The VLDB Journal
DOI 10.1007/s00778-015-0398-x

REGULAR PAPER

The matrix mechanism: optimizing linear counting queries under differential privacy

Chao Li[1,4] · **Gerome Miklau**[1] · **Michael Hay**[2] · **Andrew McGregor**[1] · **Vibhor Rastogi**[3]

Received: 26 January 2014 / Revised: 22 April 2015 / Accepted: 16 July 2015

Abstract Differential privacy is a robust privacy standard that has been successfully applied to a range of data analysis tasks. We describe the *matrix mechanism*, an algorithm for answering a workload of linear counting queries that adapts the noise distribution to properties of the provided queries. Given a workload, the mechanism uses a different set of queries, called a query strategy, which are answered using a standard Laplace or Gaussian mechanism. Noisy answers to the workload queries are then derived from the noisy answers to the strategy queries. This two-stage process can result in a more complex, correlated noise distribution that preserves differential privacy but increases accuracy. We provide a formal analysis of the error of query answers produced by the mechanism and investigate the problem of computing the optimal query strategy in support of a given workload. We show that this problem can be formulated as a rank-constrained semidefinite program. We analyze two seemingly distinct techniques proposed in the literature, whose similar behavior is explained by viewing them as instances of the matrix mechanism. We also describe an extension of the mechanism in which nonnegativity constraints are included in the derivation process and provide experimental evidence of its efficacy.

Keywords Differential privacy · Linear query · Matrix mechanism · Semidefinite program · Least squares

✉ Chao Li
cornemuse@gmail.com; chaoli@cs.umass.edu

Gerome Miklau
miklau@cs.umass.edu

Michael Hay
mhay@colgate.edu

Andrew McGregor
mcgregor@cs.umass.edu

Vibhor Rastogi
vibhor.rastogi@gmail.com

1 College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA, USA

2 Department of Computer Science, Colgate University, Hamilton, NY, USA

3 Twitter Inc., San Francisco, CA, USA

4 Present Address: Google Inc., Mountain View, CA, USA

1 Introduction

Differential privacy [12] offers participants in a dataset the compelling assurance that information released about the dataset is virtually indistinguishable whether or not their personal data are included. It protects against powerful adversaries and, in most cases, offers precise accuracy guarantees. As outlined in recent surveys [7–9], it has been applied successfully to a range of data analysis tasks and to the release of summary statistics such as contingency tables [2], histograms [19,28], and order statistics [25].

Differential privacy is achieved by introducing randomness into query answers. The original algorithm for achieving ϵ-differential privacy, commonly called the Laplace mechanism [12], returns the sum of the true answer and random noise drawn from a Laplace distribution. To achieve approximate (ϵ, δ)-differential privacy, the standard algorithm adds random noise drawn from a Gaussian distribution. In both cases, the scale of the noise distribution is determined by a property of the query called its sensitivity: roughly the maximum possible change to the query answer induced by the addition or removal of one tuple. Higher sensitivity queries are more revealing about individual tuples and must receive greater noise.

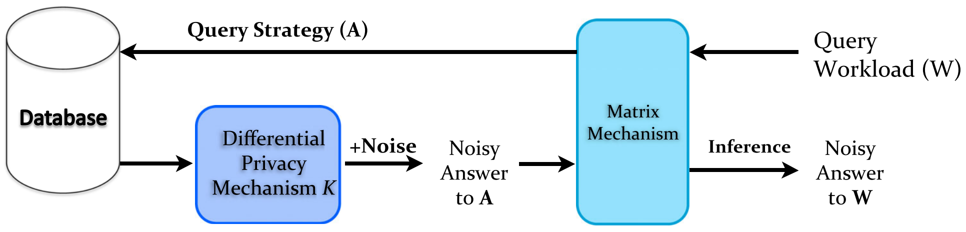

Fig. 1 Query answering using matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ with a base differentially private algorithm $\mathcal{K}$ and a strategy $\mathbf{A}$

We study batch query answering under differential privacy, in which a collection of queries (called a query workload) is submitted and answered in one interaction with the database. At the heart of our investigation is the suboptimal behavior of standard mechanisms when answers to a set of correlated queries are requested. We say two queries are correlated if the change of a tuple in the underlying database can affect both answers. Correlated workloads arise naturally in practice; however, asking correlated queries can lead to suboptimal results because correlation increases sensitivity and therefore the magnitude of the noise.

In this work, we describe the *matrix mechanism*, an improved mechanism for answering a workload consisting of linear counting queries. This class consists of queries described as a linear combination of base counts reporting the number of tuples with the given combination of attribute values. Histograms, sets of marginals, and data cubes can be expressed as workloads of linear counting queries.

As illustrated in Fig. 1, the matrix mechanism uses another private algorithm $\mathcal{K}$ (such as the Laplace or Gaussian mechanism) as a subroutine. Given a workload of queries, the matrix mechanism invokes $\mathcal{K}$ to answer a different set of queries, called a *query strategy*, and obtains noisy answers. Noisy answers to the workload queries are then derived from the noisy answers to the strategy queries. There may be more than one way to estimate a workload query from the answers to the strategy queries. In this case, the derived answer of the matrix mechanism combines the available evidence into a single consistent estimate that minimizes the variance of the noisy answer. The proof of differential privacy for the final workload query answers is straightforward because it is inherited from the privacy guarantee of $\mathcal{K}$.

Using the matrix mechanism, the noise added to the workload queries may consist of a complex, correlated noise distribution even when the underlying algorithm $\mathcal{K}$ adds independent noise to each strategy query. Such correlated noise allows for more accurate results, particularly for workloads with correlated queries.

The accuracy of the matrix mechanism depends on the query strategy selected to instantiate it. This paper explores the problem of designing the optimal strategy for a given workload. To understand the optimization problem, we first analyze the error of any query supported by a strategy. The error is determined by two essential features of the strategy: its *error profile*, a matrix which governs the distribution of error across queries, and its *sensitivity*, a scalar term that uniformly scales the error on all queries. Accurately answering a workload of queries requires choosing a strategy with a good error profile (relatively low error for the queries in the workload) and low sensitivity. We show that naive strategies typically succeed at one, but not both, of these objectives.

We then formalize the optimization problem of finding the strategy that minimizes the total error as a semidefinite program with rank constraints.

We also investigate an extension to the matrix mechanism which incorporates nonnegativity constraints into the inference process. The matrix mechanism can be seen to produce an estimate, suited to the input workload, of the original database. This estimate is represented as a vector of counts which should be nonnegative, but may include negative values due to the addition of random noise. Most works either ignore this issue or simply round negative values to zero to achieve nonnegativity. We investigate this step in more detail than prior work, showing experimentally that nonnegativity constraints can significantly reduce error when applied properly, but that they also result in a mechanism that diverges in an important way from the basic matrix mechanism: The achieved error rates are no longer independent of properties of the input database, and as a result, the formal error analysis of the mechanism cannot be carried out without taking into account properties of the input data.

After a background discussion in Sects. 2 and 3, we describe the matrix mechanism in Sect. 4. We analyze its error formally in Sect. 5 and apply our analysis to two related techniques in Sect. 6. In Sect. 7, we investigate the problem of choosing an optimal query strategy and discuss practical solutions. In Sect. 8, we study nonnegativity constraints. Finally, we discuss related work and conclude in Sects. 9 and 10.

2 Background: linear queries and query workloads

The matrix mechanism is designed to answer a set of linear queries. A linear query is an aggregation query over a single relation that can be expressed as a linear combination of a set of database counts. In this section, we first describe the representation of a relational table as a vector of nonnegative integers. We then describe linear queries, represented as vectors of coefficients, and a workload of linear queries, represented as a matrix. Lastly, we show that the matrix representation of a set of linear queries is not unique.

2.1 Data domain and cell lists

We consider a database instance I of a single-table relational schema $R(\mathbb{A})$ with attributes $\mathbb{A} = \{A_1, A_2, \ldots, A_m\}$. The domain $dom(A_i)$ of an attribute A_i may be discrete or continuous, finite or infinite, ordered or unordered. The set of all tuples that may exist in I is the cross product of the domains of attributes in $\mathbb{A}$: $dom(\mathbb{A}) = dom(A_1) \times dom(A_2) \times \cdots \times dom(A_m)$. The database instance is encoded as a vector of cell counts, each counting the number of tuples included in a distinct subset of the domain.

Definition 1 (*Cell and Cell List*) A cell is a non-empty subset of $dom(\mathbb{A})$. A cell list $\Phi = \phi_1, \ldots \phi_n$ is an ordered list of mutually exclusive cells: $\forall i, j,\ \phi_i \cap \phi_j = \emptyset$.

We do not require that the cells in a cell list cover $dom(\mathbb{A})$. For a specified cell list, a relational table can be represented (or sometimes partially represented) in the form of a *data vector* consisting of a nonnegative integer for each cell.

Definition 2 (*Data vector*) Given instance I and cell list $\Phi = \phi_1, \phi_2 \ldots \phi_n$, the vector representation of I using Φ, denoted $\mathbf{x}(I, \Phi)$, is the length-n column vector consisting of a nonnegative integer for each cell; i.e., the ith entry in $\mathbf{x}(I, \Phi)$ is $|I \cap \phi_i|$.

When I and Φ are clear from the context, we denote the data vector simply by $\mathbf{x}$.

Example 1 Consider a relational schema $R = (name, gradyear, gender, gpa)$ describing students. Figure 2a shows a sample instance of this relation. Figure 2b describes a cell list based on *gender* (male or female) and *gradyear* (2011, 2012, 2013 or 2014) where the wildcard ($*$) matches any element of the domain. Figure 2c shows the data vector that results from the instance and the cell list. Note that the sum of the counts in the data vector does not equal the total number of tuples in the instance because the cells happen not to cover the entire active domain of *gradyear*.

A common case is to define a cell list by partitioning $dom(\mathbb{A})$ according to a single ordered attribute. In this case, the data vector would describe a one-dimensional histogram. Parts of the domain that are not represented in the cell list cannot be queried. The main criterion for selecting a cell list for a given schema is that the cells support the queries of the intended workload. This can be done in multiple ways, and we return to the choice of cell lists later in this section.

2.2 Linear queries

A linear query computes a linear combination of the counts in the data vector $\mathbf{x}$.

Definition 3 (*Linear query*) A linear query is a length-n row vector $\mathbf{w} = [w_1 \ldots w_n]$ with each $w_i \in \mathbb{R}$. The answer to a linear query $\mathbf{w}$ on $\mathbf{x}$ is the dot product $\mathbf{wx} = w_1x_1 + \cdots + w_nx_n$.

Linear queries can express a variety of common aggregation queries. We refer to a linear query whose coefficients are exclusively zero or one as a *predicate counting query*, since it computes the number of tuples satisfying a predicate defined by the disjunction of the cells corresponding to query coefficients of one. For an ordered attribute domain, a *range count query* is a special case of a predicate counting query whose nonzero coefficients form a contiguous range. Range count queries have a natural extension to *multi-dimensional range count queries*. Multi-dimensional range count queries are a versatile class: Histograms, data cubes, marginal queries, and group-by queries are all sets of one-dimensional or multi-dimensional range count queries.

Although many common classes of queries are predicate counting queries, we need not restrict our attention only to linear queries with coefficients of zero or one. With other coefficients, linear queries can compute differences (e.g., query $\mathbf{w}_5$ in Fig. 3b) and can express aggregate queries that are not, strictly speaking, counting queries. For example, referring to the cell list in Fig. 2, the average graduation year of male students graduating between 2011 and 2014 can be computed as $(2011x_1 + 2012x_3 + 2013x_5 + 2014x_7)/4$.

We will consider query workloads that consist of sets of linear queries, organized into the rows of a *query matrix*.

Name	Gradyear	Gender	Gpa
Alice	2012	F	3.8
Bob	2011	M	3.1
Charlie	2014	M	3.6
Dave	2014	M	3.3
Evelyn	2013	F	3.9
Frank	2011	M	3.2
Gary	2015	M	3.5

(a) Instance of relation R

ϕ_1 :-	$R(*, 2011, M, *)$
ϕ_2 :-	$R(*, 2011, F, *)$
ϕ_3 :-	$R(*, 2012, M, *)$
ϕ_4 :-	$R(*, 2012, F, *)$
ϕ_5 :-	$R(*, 2013, M, *)$
ϕ_6 :-	$R(*, 2013, F, *)$
ϕ_7 :-	$R(*, 2014, M, *)$
ϕ_8 :-	$R(*, 2014, F, *)$

(b) Cell list Φ

x_1:	2
x_2:	0
x_3:	0
x_4:	1
x_5:	0
x_6:	1
x_7:	2
x_8:	0

(c) $\mathbf{x}$

Fig. 2 For schema $R = (name, gradyear, gender, gpa)$. **a** shows a sample instance. A cell list consisting of 8 cells described in terms of the tuples that match conditions on *gradyear* and *gender* is shown in (**b**). The database vector, shown in (**c**), accordingly consists of 8 counts

Fig. 3 **a** A query matrix $\mathbf{W}$ consisting of five linear queries, **b** the description of the queries in $\mathbf{W}$ using the cell list Φ in Fig. 2, **c** the evaluation of $\mathbf{W}$ on $\mathbf{x}$, **d** a semantically equivalent query matrix $\mathbf{W}'$ expressed w.r.t. a reduced cell list (columns 5 and 6 in $\mathbf{W}$ have been combined to get $\mathbf{W}'$)

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \end{bmatrix}$$

(a) Query matrix $\mathbf{W}$

$\mathbf{w}_1$: Students of any gender with $gradyear \in [2011, 2014]$
$\mathbf{w}_2$: Students with $gradyear \in [2011, 2012]$
$\mathbf{w}_3$: Female students with $gradyear \in [2011, 2012]$
$\mathbf{w}_4$: Male students with $gradyear \in [2011, 2012]$
$\mathbf{w}_5$: Difference between 2013 grads and 2014 grads

(b) Five linear queries

$$\begin{aligned} \mathbf{w}_1\mathbf{x} &= x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 &&= 6 \\ \mathbf{w}_2\mathbf{x} &= x_1 + x_2 + x_3 + x_4 &&= 3 \\ \mathbf{w}_3\mathbf{x} &= x_2 + x_4 &&= 1 \\ \mathbf{w}_4\mathbf{x} &= x_1 + x_3 &&= 2 \\ \mathbf{w}_5\mathbf{x} &= x_5 + x_6 - x_7 - x_8 &&= -1 \end{aligned}$$

(c) The evaluation of $\mathbf{Wx}$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix}$$

(d) Query matrix $\mathbf{W}'$

Definition 4 (*Query matrix*) A query matrix is a collection of m linear queries, arranged by rows to form an $m \times n$ matrix.

If $\mathbf{W}$ is an $m \times n$ query matrix, the evaluation of $\mathbf{W}$ results in a length-m column vector of query answers, which can be computed as the matrix product $\mathbf{Wx}$.

Example 2 Figure 3 shows a query matrix representing a workload of five linear queries, along with the meaning of the queries using the cell list in Fig. 2b. The queries are evaluated by computing $\mathbf{Wx}$, as shown in Fig. 3c.

As the example above shows, a linear query is simply a vector of coefficients, whereas the cell lists provide a semantics for the query in terms of the schema for a particular relation.

2.3 Representing query workloads in matrix form

Later in the paper, we will assume that an analyst has decided on a workload of queries of interest, selected a cell list, and represented the workload as a query matrix, which is the main input to our algorithms. We describe next a few guidelines and subtleties involved in representing a query workload in matrix form.

The matrix mechanism can be seen as automatically optimizing the workload to reduce error. As a result, the analyst does not have to think carefully about the workings of the privacy mechanism when representing the workload. In particular, the analyst needs not try to reduce the sensitivity of the workload or avoid subtle redundancies in queries. Instead, the analyst should include in the workload *all* queries of interest, even if some queries could be computed from others in the workload. As a concrete example, in Fig. 3b, $\mathbf{w}_4$ can be computed as $(\mathbf{w}_2 - \mathbf{w}_3)$, but it is nevertheless included in the workload. This reflects our assumption that the goal is to simultaneously answer all given workload queries with minimum aggregate error, treating each equally. Duplicate queries should be removed from the workload; however, individual rows of the workload may be multiplied by a positive scalar value. This has the effect of increasing the importance of the query and reducing the error of that query relative to total error of the workload.

After deciding on the workload queries, the next step is to select an appropriate cell list that can support the workload queries. If each attribute domain is finite, then it is possible to fully represent instance I by defining the (finite) vector $\mathbf{x}$ with one cell for every element of $dom(\mathbb{A})$. Then, $\mathbf{x}$ is a bit vector of size $|dom(\mathbb{A})|$ with nonzero counts for each tuple present in I. This is also a vector representation of the full contingency table built from I (note that if the schema contains infinite attribute domains, they would typically be partitioned into finite regions of sufficient granularity to support the desired queries).

Selecting the cell list in this manner allows a wide range of desired queries to be supported. But it is often inefficient, since the size of $\mathbf{x}$ then grows exponentially with the sizes of the attribute domains, and ineffective, since the base counts are typically too small to be estimated very accurately. Alternatively, it may be sufficient to partially represent I by the cell counts in $\mathbf{x}$, for example, by focusing on a subset of the attributes of $\mathbb{A}$ that are relevant to a specialized set of queries and/or a subset of the attribute domains (as in Example 1).

There are many ways to represent semantically equivalent workloads in matrix form. First of all, any workload matrix implies a particular order for the workload queries, but this order is irrelevant to the set-based semantic of a query workload. Further, there are many feasible choices for the cell list supporting a given workload, and equivalent workloads can be expressed over alternative cell lists:

Definition 5 (*Workload semantic equivalence*) Workload $\mathbf{W}$ over cell list Φ is semantically equivalent to workload $\mathbf{W}'$ over cell list Φ', denoted $(\mathbf{W}, \Phi) \equiv (\mathbf{W}', \Phi')$, if there is a permutation matrix $\mathbf{P}$ such that for every instance I, $\mathbf{Wx}(I, \Phi) = \mathbf{PW}'\mathbf{x}(I, \Phi')$.

Example 3 Observe in Fig. 3 that columns 5 and 6 of workload $\mathbf{W}$ are identical. With respect to the example workload, positions 5 and 6 of the data vector are either both ignored, or summed together. It follows that cells ϕ_5 and ϕ_6 can be combined and the query matrix altered by dropping one of the columns and that these operations will not modify the semantics of the workload. More precisely, $(\mathbf{W}, \Phi) \equiv (\mathbf{W}', \Phi')$ where Φ' is derived from Φ as follows. The first four cells in Φ' are equal to those of Φ, cell $\phi'_5 = \phi_5 \vee \phi_6$, $\phi'_6 = \phi_7$, and $\phi'_7 = \phi_8$. Observe that $\mathbf{W}'$ results from removing column 6 from $\mathbf{W}$.

The following proposition shows that semantic equivalence can be characterized by considering a small set of semantics-preserving operations over cell lists and workload matrices.

Proposition 1 *For workload* $\mathbf{W}$ *over cell list* Φ *and a workload* $\mathbf{W}'$ *over cell list* Φ', $(\mathbf{W}, \Phi) \equiv (\mathbf{W}', \Phi')$ *if and only if* $\mathbf{W}'$ *and* Φ' *result from a sequence of one or more of the following operations:*

1. Permutation *apply permutation μ to the rows of* $\mathbf{W}$*, or the cells of Φ and the columns of* $\mathbf{W}$.
2. Cell union *if* $\mathbf{W}$ *contains two columns with identical coefficients, form* $\mathbf{W}'$ *by removing one of the columns and replacing the cells by their union.*
3. Cell division *for any column W_i of* $\mathbf{W}$ *and corresponding cell ϕ_i of Φ, construct Φ' by replacing condition ϕ_i with ϕ_{i_1} and ϕ_{i_2} where $\phi_{i_1} \cup \phi_{i_2} = \phi_i$ and $\phi_{i_1} \cap \phi_{i_2} = \emptyset$. Then, associate cell ϕ_{i_1} and ϕ_{i_2} with the column of coefficients W_i (i.e., two copies of W_i will appear in* $\mathbf{W}'$*).*
4. Add irrelevant cells *add a new cell to Φ and a corresponding column to* $\mathbf{W}$ *whose coefficients are all zeros.*
5. Remove irrelevant cells *if* $\mathbf{W}$ *contains a column of zeros, remove it along with its associated cell in Φ.*

Proof The sufficiency of the conditions in Proposition 1 is easily verified. Here, we prove the necessity of the conditions.

Given $\mathbf{W}_1$ over Φ_1 and $\mathbf{W}_2$ over Φ_2 such that $(\mathbf{W}_1, \Phi_1) \equiv (\mathbf{W}_2, \Phi_2)$. As it is defined in Definition 5, there exists a permutation matrix $\mathbf{P}$ such that $\mathbf{W}_1\mathbf{x}(I, \Phi_1) = \mathbf{P}\mathbf{W}_2\mathbf{x}(I, \Phi_2)$ for any instance I. Noticing the row permutation is a part of the Permutation operation in Proposition 1, it is sufficient to consider the case that $\mathbf{P} = \mathbf{I}$.

Further, given a workload $\mathbf{W}_1$ over Φ_1, construct a workload $\mathbf{W}'_1$ over Φ'_1 by performing operation 2 and 5 until the workload contains neither duplicate column nor column of zeros. Then, $(\mathbf{W}_1, \Phi_1) \equiv (\mathbf{W}'_1, \Phi'_1)$ according to the sufficiency of those conditions. Apply the same process to $\mathbf{W}_2$ to get the workload $\mathbf{W}'_2$ over Φ'_2 such that $(\mathbf{W}'_2, \Phi'_2) \equiv (\mathbf{W}_2, \Phi_2)$. According to Definition 5, $(\mathbf{W}'_1, \Phi'_1)$ must be semantically equivalent to $(\mathbf{W}'_2, \Phi'_2)$.

First of all, $\bigvee_{\phi \in \Phi'_1} \phi = \bigvee_{\phi \in \Phi'_2} \phi$. Otherwise, without loss of generality, assume that $\bigvee_{\phi \in \Phi'_1} \phi$ is not a subset of $\bigvee_{\phi \in \Phi'_2} \phi$ and let

$$I_0 = \left\{t | \phi(t) \wedge (\neg\phi'(t)) \text{ is True}, \forall \phi \in \Phi'_1, \forall \phi' \in \Phi'_2\right\}.$$

Then, $I_0 \neq \emptyset$ and $\mathbf{W}'_1\mathbf{x}(I_0, \Phi'_1) \neq \mathbf{W}'_2\mathbf{x}(I_0, \Phi'_2) = \mathbf{0}$, which contradicts with the fact that $(\mathbf{W}'_1, \Phi'_1) \equiv (\mathbf{W}'_2, \Phi'_2)$.

In addition, for any i, j such that $\phi_i \in \Phi'_1$ and $\phi'_j \in \Phi'_2$ such that $\phi_i \wedge \phi'_j \neq \emptyset$. Let W_i be the column of $\mathbf{W}'_1$ corresponding to ϕ_i and W'_j be the column of $\mathbf{W}'_2$ corresponding to ϕ'_j. W_i must be equal to W'_j. Otherwise, let $I_1 = \{t | \phi_i(t) \wedge \phi'_j(t) \text{ is True}\}$ and $\mathbf{W}'_1\mathbf{x}(I_1, \Phi'_1) = |I_1|W_i \neq \mathbf{W}'_2\mathbf{x}(I_1, \Phi'_2) = |I_1|W'_j$, which leads to a contradiction. Moreover, since neither $\mathbf{W}'_1$ nor $\mathbf{W}'_2$ contains duplicate columns, any cell condition in Φ'_1 other than ϕ_i is disjoint with ϕ'_j and any cell condition in Φ'_2 other than ϕ'_j is disjoint with ϕ_i. Therefore, $\phi_i = \phi'_j$; otherwise, $\bigvee_{\phi \in \Phi'_1} \phi \neq \bigvee_{\phi \in \Phi'_2} \phi$.

Above all, we know there must exist a permutation μ to the cells of Φ'_1 and the columns of $\mathbf{W}'_1$ that gets us $(\mathbf{W}'_2, \Phi'_2)$. Thus, $(\mathbf{W}_1, \Phi_1)$ can be transformed into $(\mathbf{W}_2, \Phi_2)$ with the operations in Proposition 1. □

We will show later in the paper that many aspects of the performance of our algorithms are independent of the cell list used and the particular query matrix that results. Most importantly, the optimal error achievable for a workload is the same for any semantically equivalent workload matrix. However, in terms of efficiency, it is beneficial to represent a workload with the smallest possible set of cells. The number of cells in the cell list, n, (which is also the number of columns in the workload matrix) is a key parameter in the computational complexity of the algorithms to come. Fortunately, using Proposition 1, it is straightforward to create the smallest cell list for a given workload of interest. After starting with any feasible representation of the workload, we can repeatedly apply steps (2) and (5), in any order.

3 Background: differential privacy

Informally, a randomized algorithm is differentially private if it produces statistically close outputs whether or not any one individual's record is present in the database. Two instances I and I' are neighbors, denoted nbrs(I, I') if they differ by at most one record, i.e., if $|(I - I') \cup (I' - I)| = 1$.

Definition 6 (*Differential privacy*) A randomized algorithm $\mathcal{K}$ is (ϵ, δ)-differentially private if for any instances I, I' such that nbrs(I, I'), and any subset of outputs $S \subseteq Range(\mathcal{K})$, the following holds:

$$Pr[\mathcal{K}(I) \in S] \leq \exp(\epsilon) \times Pr[\mathcal{K}(I') \in S] + \delta,$$

where the probability is taken over the randomness of the $\mathcal{K}$.

If an algorithm satisfies the definition above for $\delta = 0$, then it is ϵ-differentially private. When $\delta > 0$, the privacy standard is sometimes referred to as *approximate differential privacy*. The matrix mechanism can be used with both definitions.

Both ϵ- and (ϵ, δ)-differential privacy can be satisfied by algorithms that add random noise to query answers. The magnitude of the required noise is determined by the privacy parameters, ϵ and/or δ, and the *sensitivity* of the set of queries: the maximum change in a vector of query answers over any two neighboring databases. The two privacy definitions differ, however, in the measurement of sensitivity and in their noise distributions. Standard ϵ-differential privacy can be achieved by adding Laplace noise calibrated to the L_1 sensitivity of the queries [12]. Approximate (ϵ, δ)-differential privacy can be achieved by adding Gaussian noise calibrated to the L_2 sensitivity of the queries [10,22].

Since our query workloads are represented as matrices, we describe the sensitivity of a workload matrix as a matrix norm. Recall that, for any cell list Φ, cells are always disjoint and $\mathbf{x}(I, \Phi)$ is the vector representation of I using Φ. Since neighboring databases I and I' differ in exactly one tuple, it follows that the corresponding vectors $\mathbf{x}(I, \Phi)$ and $\mathbf{x}(I', \Phi)$ differ in at most one component, by at most one.

In the propositions below, cols$(\mathbf{W})$ is the set of column vectors W_i of $\mathbf{W}$. For a query matrix $\mathbf{W}$, the L_1 sensitivity is the maximum L_1 norm of the columns of $\mathbf{W}$.

Proposition 2 (L_1 Query matrix sensitivity) *For any cell list Φ, the L_1 sensitivity of a query matrix $\mathbf{W}$ using cell list Φ is denoted $\|\mathbf{W}\|_1$ and defined as:*

$$
\begin{aligned}
||\mathbf{W}||_1 &\overset{\text{def}}{=} \max_{I,I' \in nbrs(I,I')} \left\| \mathbf{W}\mathbf{x}(I, \Phi) - \mathbf{W}\mathbf{x}(I', \Phi) \right\|_1 \\
&= \max_{W_i \in cols(\mathbf{W})} \|W_i\|_1
\end{aligned}
$$

Similarly, the L_2 sensitivity of $\mathbf{W}$ is equal to the maximum L_2 norm of the columns of $\mathbf{W}$.

Proposition 3 (L_2 Query matrix sensitivity) *For any cell list Φ, the L_2 sensitivity of a query matrix $\mathbf{W}$ using cell list Φ is denoted $||\mathbf{W}||_2$ and defined as:*

$$
\begin{aligned}
||\mathbf{W}||_2 &\overset{\text{def}}{=} \max_{I,I' \in nbrs(I,I')} ||\mathbf{W}\mathbf{x}(I, \Phi) - \mathbf{W}\mathbf{x}(I', \Phi)||_2 \\
&= \max_{W_i \in cols(\mathbf{W})} ||W_i||_2
\end{aligned}
$$

It is clear from the above propositions that the sensitivity of a query matrix is in fact independent of any cell list that accompanies it and our notation reflects this. Further, we occasionally use $||\mathbf{W}||$ (without a subscript) to represent the sensitivity when the context does not specify whether it is L_1 or L_2 sensitivity.

$$
\underset{\mathbf{I}_4}{\begin{bmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{bmatrix}}
\qquad
\underset{\mathbf{H}_4}{\begin{bmatrix} 1&1&1&1\\ 1&1&0&0\\ 0&0&1&1\\ 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{bmatrix}}
\qquad
\underset{\mathbf{Y}_4}{\begin{bmatrix} 1&1&1&1\\ 1&1&-1&-1\\ 1&-1&0&0\\ 0&0&1&-1 \end{bmatrix}}
$$

Fig. 4 Query matrices with a cell of size four. Each is full rank. $\mathbf{I}_4$ returns a count for each cell. $\mathbf{H}_4$ computes seven sums, hierarchically partitioning the cells. $\mathbf{Y}_4$ is based on the Haar wavelet

Example 4 Figure 4 shows three query matrices, over an unspecified cell list of size four, which we use as a running example. $\mathbf{I}_4$ is the identity matrix of size four. This matrix consists of four queries, each asking for an individual element of the data vector $\mathbf{x}$. $\mathbf{H}_4$ contains seven queries, which represent a binary hierarchy of sums: The first row is the sum of the elements of $\mathbf{x}$, the second and third rows each sum one half of $\mathbf{x}$, and the last four rows return individual elements of $\mathbf{x}$. $\mathbf{Y}_4$ is the matrix of the Haar wavelet. It can also be seen as a hierarchical set of queries: The first row is the total sum, the second row computes the difference between sums in two halves of $\mathbf{x}$, and the last two rows return differences between smaller partitions of $\mathbf{x}$.

The sensitivity of each of the query matrices in Fig. 4 is: $||\mathbf{I}_4||_1 = 1$ and $||\mathbf{H}_4||_1 = ||\mathbf{Y}_4||_1 = 3$; $||\mathbf{I}_4||_2 = 1$ and $||\mathbf{H}_4||_2 = ||\mathbf{Y}_4||_2 = \sqrt{3}$. A change by one in any component $\mathbf{x}_i$ will change the query answer $\mathbf{I}_4\mathbf{x}$ by exactly one under both L_1 and L_2, but will change $\mathbf{H}_4\mathbf{x}$ and $\mathbf{Y}_4\mathbf{x}$ by 3 under L_1 and $\sqrt{3}$ under L_2 since each x_i contributes to three predicate queries in both $\mathbf{H}_4$ and $\mathbf{Y}_4$.

The following propositions describe, in vector form, the standard mechanisms for answering a set of queries under ϵ-differential privacy and (ϵ, δ)-differential privacy. The Laplace mechanism [7,10] achieves ϵ-differential privacy by adding Laplace noise calibrated to the L_1 sensitivity of the input queries. We use Laplace$(b)^m$ to denote a column vector consisting of m independent samples drawn from a Laplace distribution with mean 0 and scale b.

Proposition 4 (Laplace mechanism) *Given an $m \times n$ query matrix $\mathbf{W}$, the randomized algorithm $\mathcal{L}$ that outputs the following vector is ϵ-differentially private:*

$$\mathcal{L}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + Laplace(b)^m$$

where $b = ||\mathbf{W}||_1/\epsilon$.

The Gaussian mechanism [22] achieves (ϵ, δ)-differential privacy by adding Gaussian noise calibrated to the L_2 sensitivity. We use Normal$(\sigma)^m$ to denote a column vector consisting of m independent samples drawn from a Gaussian distribution with mean 0 and scale σ.

Proposition 5 (Gaussian mechanism [10,22]) *Given an $m \times n$ query matrix $\mathbf{W}$, the randomized algorithm $\mathcal{G}$ that outputs the following vector is (ϵ, δ)-differentially private:*

$$\mathcal{G}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + Normal(\sigma)^m$$

where $\sigma = ||\mathbf{W}||_2 \sqrt{2\ln(2/\delta)}/\epsilon$.

Recall that $\mathbf{Wx}$ is a vector consisting of the true answers to each query in $\mathbf{W}$. The algorithms above add independent Laplace noise (scaled by $||\mathbf{W}||_1$ and ϵ) or Gaussian noise (scaled by $||\mathbf{W}||_2$, ϵ, and δ) to each query answer. Thus, both $\mathcal{L}(\mathbf{W}, \mathbf{x})$ and $\mathcal{G}(\mathbf{W}, \mathbf{x})$ are length-m column vectors containing a noisy answer for each linear query in $\mathbf{W}$. The vector form of the Laplace and Gaussian mechanisms above incorporates what is sometimes referred to as *parallel composition* for answering multiple queries [23].

4 The matrix mechanism

In this section, we present the formal basis for the matrix mechanism. Given a workload of linear queries, the matrix mechanism uses an alternative set of queries, the strategy, which are answered privately by a standard mechanism. Answers to the workload queries are then derived from the strategy queries. Below, we define the set of queries whose estimates can be derived from the strategy and we provide optimal mechanisms for deriving estimates. In the remainder of this paper, we use $\mathbf{W}$ and $\mathbf{A}$ to denote the query workload and query strategy as well as their matrix representation.

Given a query strategy $\mathbf{A}$ and its noisy answer from any differentially private algorithm, in order to answer a query workload $\mathbf{W}$ with the answer to $\mathbf{A}$, each query $\mathbf{w}$ in $\mathbf{W}$ should be able to be expressed as a linear combination of queries in $\mathbf{A}$:

Definition 7 (*Support*) Given a query workload $\mathbf{W}$ and a query strategy $\mathbf{A}$, we say $\mathbf{A}$ supports $\mathbf{W}$ if each query in $\mathbf{W}$ can be expressed as a linear combination of queries in $\mathbf{A}$. In other words, there exists a solution matrix $\mathbf{X}$ to the linear system $\mathbf{W} = \mathbf{XA}$.

To derive the answer to $\mathbf{W}$, one needs to solve the linear system $\mathbf{W} = \mathbf{XA}$. There may be multiple solutions to the linear system, so we take the advantage of the uniqueness of the Moore–Penrose pseudoinverse of matrix $\mathbf{A}$ and express the answer to $\mathbf{W}$ as follows:

Definition 8 (*Workload derivation*) Let $\mathbf{A}$ be a query strategy that supports $\mathbf{W}$ and $\hat{\mathbf{y}}$ be noisy answers to $\mathbf{A}$. Then, the derived noisy answer to $\mathbf{W}$ is defined as $\mathbf{WA}^+\hat{\mathbf{y}}$, where $\mathbf{A}^+$ is the Moore–Penrose pseudoinverse of matrix $\mathbf{A}$.

In particular, when $\mathbf{A}$ is a full rank matrix, $\mathbf{A}^+\hat{\mathbf{y}}$ is the estimate of $\mathbf{x}$ that minimizes the squared error given the noisy observations of the strategy queries.

Example 5 Recall the cell conditions and queries in Fig. 2. Let the query workload be $\mathbf{W}_1 = \{\mathbf{q}_2, \mathbf{q}_3\}$. Then, query strategy $\mathbf{A}_1 = \{\mathbf{q}_1, \mathbf{q}_2\}$ does not support $\mathbf{W}$ since it cannot represent $\mathbf{q}_3$. $\mathbf{A}_2 = \{\mathbf{q}_3, \mathbf{q}_4\}$ supports $\mathbf{W}_1$ and

$$\mathbf{WA}_2^+ = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

In this case, the answer to $\mathbf{W}_1$ can be uniquely computed from the answer to $\mathbf{A}_2$. Further, the rows of $\mathbf{WA}_2^+$ show exactly the linear combinations of the answers to $\mathbf{A}_2$ that are used to compute the answer to $\mathbf{W}_1$.

Now, we describe the matrix mechanism. Given any differentially private algorithm $\mathcal{K}$ that answers linear queries, the matrix mechanism can be considered as an extension of $\mathcal{K}$. When instantiated with the supporting query strategy $\mathbf{A}$, the matrix mechanism is denoted as $\mathcal{M}_{\mathcal{K},\mathbf{A}}$.

Definition 9 (*Matrix mechanism*) Given an $m \times n$ workload matrix $\mathbf{W}$, a $p \times n$ strategy matrix $\mathbf{A}$ that supports $\mathbf{W}$ and a differentially private algorithm $\mathcal{K}(\mathbf{A}, \mathbf{x})$ that answers $\mathbf{A}$ with a given database instance $\mathbf{x}$. The matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ outputs the following vector:

$$\mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{W}, \mathbf{x}) = \mathbf{WA}^+\mathcal{K}(\mathbf{A}, \mathbf{x}). \tag{1}$$

Recall Fig. 1 illustrating the process of query answering using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$. Given query workload $\mathbf{W}$, the matrix mechanism uses a query strategy $\mathbf{A}$ that supports $\mathbf{W}$, answers $\mathbf{A}$ with the differentially private algorithm $\mathcal{K}$, and outputs the derived answer to $\mathbf{W}$ using the answer to $\mathbf{A}$. The power of the matrix mechanism comes from the possibility that query strategy $\mathbf{A}$ can be carefully designed or optimized. Note that the derivation of the workload queries uses only the answers to the strategy queries, with no additional access to the database $\mathbf{x}$. The matrix mechanism therefore inherits the privacy properties and unbiasedness of $\mathcal{K}$.

Proposition 6 *The matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ inherits the privacy guarantee of $\mathcal{K}$ and is unbiased if $\mathcal{K}$ is unbiased.*

In general, Eq. (1) is valid for any differentially private mechanism $\mathcal{K}$. However, the choice of $\mathcal{K}$ impacts the difficulty of the error analysis and the complexity of finding a query strategy $\mathbf{A}$ to minimize the error for a given workload $\mathbf{W}$. In most of this paper, we focus on differentially private algorithms that are data independent:

Definition 10 A differentially private algorithm $\mathcal{K}(\mathbf{A}, \mathbf{x})$ is data independent if it can be represented as

$$\mathcal{K}(\mathbf{A}, \mathbf{x}) = \mathbf{Ax} + \tilde{\mathbf{b}}_{\mathbf{A}},$$

where $\tilde{\mathbf{b}}_{\mathbf{A}}$ is a random vector that is independent of the choice of $\mathbf{x}$.

In addition, the algorithm $\mathcal{K}(\mathbf{A}, \mathbf{x})$ we are considering should also satisfy the following two properties: (1) The algorithm should add independent noise to each query of $\mathbf{A}$, (2) the standard deviation of noise added is linearly scaled up with $||\mathbf{A}||$. Analytically, such an algorithm $\mathcal{K}$ can be represented into the following form:

$$\mathcal{K}(\mathbf{A}, \mathbf{x}) = \mathbf{A}\mathbf{x} + ||\mathbf{A}||\tilde{\mathbf{b}}, \tag{2}$$

where $\tilde{\mathbf{b}}$ is a vector of i.i.d random variables that does not depend on $\mathbf{W}$ or $\mathbf{x}$. Many data-independent differentially private mechanisms based on noise adding can be represented in the form of Eq. (2), such as Laplace mechanism [10,12], Gaussian mechanism [12,22], geometric mechanism [14], and K-norm mechanism [18].

Proposition 7 *When $\mathcal{K}$ has the form of Eq. (2), the matrix mechanism can be presented as:.*

$$\mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{W}\mathbf{A}^{+}||\mathbf{A}||\tilde{\mathbf{b}}. \tag{3}$$

In the rest of the paper, we focus on the matrix mechanism with the form of Eq. (3). In particular, we use the ϵ-matrix mechanism to denote the case in which $\mathcal{K}$ is Laplace mechanism, and the (ϵ, δ)-matrix mechanism to denote the case in which $\mathcal{K}$ is Gaussian mechanism. In this case, Eq. 3 can be readily compared with the Laplace and Gaussian mechanisms from Propositions 4 and 5. Both the Laplace and Gaussian mechanisms add to $\mathbf{W}\mathbf{x}$ a vector of independently sampled noise that is calibrated to the sensitivity of $\mathbf{W}$, while the matrix mechanism adds a more complex noise vector (independent noise $\tilde{\mathbf{b}}$ transformed by $\mathbf{W}\mathbf{A}^{+}$) and one that is calibrated to $||\mathbf{A}||$, the sensitivity of the strategy.

According to properties of the Moore–Penrose pseudoinverse (see Proposition 17 in the appendix), since entries of $||\mathbf{A}||\tilde{\mathbf{b}}$ are generated from i.i.d random distributions, $\mathbf{W}\mathbf{A}^{+}$ is the min-variance estimation to the noisy answer of $\mathbf{A}$.

Proposition 8 *When $\mathcal{K}$ has the form of Eq. (2), the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ produces the min-variance estimator to* $\mathbf{W}\mathbf{x}$ *given* $\mathcal{K}(\mathbf{A}, \mathbf{x})$.

4.1 Running time of the matrix mechanism

Here, we briefly describe how to use the matrix mechanism and its running time. To run the matrix mechanism, the user should first fix a basic differentially private algorithm $\mathcal{K}$ like the Laplace or Gaussian mechanism. For each given workload $\mathbf{W}$, the matrix mechanism can be deployed in four steps:

1. Strategy selection given $\mathbf{W}$: It is non-trivial to generate optimal strategies for an arbitrary workload $\mathbf{W}$. The complexity of optimal strategy selection will be discussed in Sect. 7. On the other hand, for certain workloads, the strategy can be set to known strategies proposed in the literature [19,28].
2. Answering the strategy $\mathbf{A}$: All queries in the strategy $\mathbf{A}$ on the dataset $\mathbf{x}$ must be answered using $\mathcal{K}$, which takes $O(m_1 \times n)$ time for an $m_1 \times n$ strategy $\mathbf{A}$.
3. Computing $\mathbf{W}\mathbf{A}^{+}$: In general, computing $\mathbf{W}\mathbf{A}^{+}$ for an $m \times n$ workload $\mathbf{W}$ and an $m_1 \times n$ strategy $\mathbf{A}$ takes $O(mm_1n)$ time.
4. Answering the workload $\mathbf{W}$: The answer to the workload $\mathbf{W}$ can be computed by multiplying the matrix $\mathbf{W}\mathbf{A}^{+}$ by the vector of answers to the strategy $\mathbf{A}$, which takes $O(mn)$ time for an $m \times n$ workload $\mathbf{W}$.

It is important to note that, for any fixed workload, one only needs to run steps 1 and 3 *once*. Their results can be used repeatedly on any input database. Therefore, for a fixed workload, one can precompute steps 1 and 3 so that the running time of the matrix mechanism can be reduced to $O((m + m_1)n)$. When $m_1 = O(m)$, this running time is asymptotically the same as the time for answering a general $m \times n$ workload using $\mathcal{K}$ (the standard Gaussian or Laplace mechanism), which takes $O(mn)$ time.

The efficiency of the matrix mechanism can be further improved when special strategy matrices are used. In particular, the instances of the matrix mechanism described in [2,6,19,28,31] each focus on workloads of range queries and use special strategy matrices for which faster evaluation is possible: It only takes $O(n)$ time to deploy the matrix mechanism in these cases.

5 Analyzing the error of the matrix mechanism

The error introduced using the matrix mechanism is impacted by two factors: the noise from the differentially private mechanism $\mathcal{K}$ and the linear combinations that generate the answers to the workload queries $\mathbf{W}$ from the answers to the strategy queries $\mathbf{A}$. We analyze the error of the matrix mechanism in this section and derive a closed-form expression for given $\mathcal{K}$, $\mathbf{W}$, and $\mathbf{A}$.

Along with the error analysis, we investigate another kind of equivalence among workloads, the total error equivalence. Such workloads are semantically different, but always have the same error under the matrix mechanism. Further, we study the equivalence among strategies and demonstrate that strategies consisting of different sets of queries are fully exchangeable under the matrix mechanism.

5.1 Error of the matrix mechanism

Given a query $\mathbf{w}$ and a query strategy $\mathbf{A}$ that supports $\mathbf{w}$, the error of answering $\mathbf{w}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is defined as the mean-squared error (variance) of the estimated answer to $\mathbf{w}$.

Definition 11 (*Error of a single query*) Let $\mathbf{x}$ be the database instance and $\mathbf{A}$ be a query strategy. Given a single query $\mathbf{w}$ that $\mathbf{A}$ supports, the error of answer $\mathbf{w}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is:

$$\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{w}) = \mathbb{E}\left[\left(\mathbf{wx} - \mathbf{wA}^{+}\mathcal{K}(\mathbf{A},\mathbf{x})\right)^2\right].$$

For a query workload $\mathbf{W}$ that $\mathbf{A}$ supports, the total error of answering $\mathbf{W}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is:

$$\textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) = \sum_{\mathbf{w}\in\mathbf{W}} \mathbb{E}\left[\left(\mathbf{wx} - \mathbf{wA}^{+}\mathcal{K}(\mathbf{A},\mathbf{x})\right)^2\right].$$

For a query strategy $\mathbf{A}$, the following proposition describes how to compute the error of answering a supported query $\mathbf{w}$ or a supported query workload $\mathbf{W}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$. The Frobenius norm, $||\cdot||_F$, used below is the square root of the sum of squares of all entries in a vector/matrix.

Proposition 9 *Let* $\mathbf{A}$ *be a query strategy. Given a query* $\mathbf{w}$ *that* $\mathbf{A}$ *supports, the error of answering* $\mathbf{w}$ *using the matrix mechanism* $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ *is:*

$$\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{w}) = P(\mathcal{K})||\mathbf{A}||^2||\mathbf{wA}^{+}||_F^2. \tag{4}$$

For a query workload $\mathbf{W}$ *that* $\mathbf{A}$ *supports, the total error of answering* $\mathbf{W}$ *using the matrix mechanism* $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ *is:*

$$\textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) = P(\mathcal{K})||\mathbf{A}||^2||\mathbf{WA}^{+}||_F^2. \tag{5}$$

Here, $P(\mathcal{K})$ *is a constant determined by* $\mathcal{K}$ *and independent of* $\mathbf{W}$, $\mathbf{A}$, *and* $\mathbf{x}$.

Proof Recall that $\mathcal{K}(\mathbf{A},\mathbf{x}) = \mathbf{Ax} + ||\mathbf{A}||\tilde{\mathbf{b}}$ and the entries of $\tilde{\mathbf{b}}$ are i.i.d random variables. Let $\tilde{\mathbf{b}} = (b_1,\ldots,b_n)$. According to Definition 11, the error of answer $\mathbf{w}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is:

$$\begin{aligned}\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{w}) &= \mathbb{E}\left[\left(\mathbf{wx} - \mathbf{wA}^{+}\mathcal{K}(\mathbf{A},\mathbf{x})\right)^2\right]\\ &= \mathbb{E}\left[\left(\mathbf{wx} - \mathbf{wA}^{+}(\mathbf{Ax} + ||\mathbf{A}||\tilde{\mathbf{b}})\right)^2\right]\\ &= \mathbb{E}\left[(\mathbf{wA}^{+}||\mathbf{A}||\tilde{\mathbf{b}})^2\right]\\ &= \mathrm{Var}(\mathbf{wA}^{+}||\mathbf{A}||\tilde{\mathbf{b}})\\ &= \mathrm{Var}(b_1)||\mathbf{A}||^2||\mathbf{wA}^{+}||_F^2.\end{aligned}$$

Therefore, for a given query workload $\mathbf{W}$,

$$\begin{aligned}\textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) &= \sum_{\mathbf{w}\in\mathbf{W}} \mathbb{E}[(\mathbf{wx} - \mathbf{wA}^{+}\mathcal{K}(\mathbf{A},\mathbf{x}))^2]\\ &= \mathrm{Var}(b_1)\sum_{\mathbf{w}\in\mathbf{W}} ||\mathbf{A}||^2||\mathbf{wA}^{+}||_2^2\\ &= \mathrm{Var}(b_1)||\mathbf{A}||^2||\mathbf{WA}^{+}||_F^2.\end{aligned}$$

Let $P(\mathcal{K}) = \mathrm{Var}(b_1)$, and recall that $\tilde{\mathbf{b}}$ only depends on $\mathcal{K}$. $P(\mathcal{K})$ is independent of $\mathbf{W}$, $\mathbf{A}$, and $\mathbf{x}$. □

Since we only consider the matrix mechanism based on data-independent differentially private algorithms, the results in Proposition 9 do not contain the database instance $\mathbf{x}$. Recall that the parameter $P(\mathcal{K})$ is a constant determined by the given private algorithm $\mathcal{K}$. In particular, $P(\mathcal{K}) = 2/\epsilon^2$ and $P(\mathcal{K}) = 2\log(1/\delta)/\epsilon^2$ when $\mathcal{K}$ is the Laplace mechanism and Gaussian mechanism, respectively. Moreover, the computation of $||\mathbf{A}||$ is also determined by the choice of $\mathcal{K}$: It can either be the maximum L_1 or L_2 norm of the columns of $\mathbf{A}$ depending on whether $\mathcal{K}$ satisfies ϵ- or (ϵ,δ)-differential privacy, respectively.

Notice that a query strategy $\mathbf{A}$ impacts both Eqs. (4) and (5) in two ways: $||\mathbf{A}||$ and $||\mathbf{WA}^{+}||_F$. The former determines the cost of answering $\mathbf{A}$ using $\mathcal{K}$, and the latter reflects the hardness of computing the answer to $\mathbf{W}$ from the answer to $\mathbf{A}$. To achieve minimum error, the ideal query strategy $\mathbf{A}$ should have low sensitivity while being as similar to $\mathbf{W}$ as possible. Consider two extreme cases for the choice of A. First, if $\mathbf{A} = \mathbf{I}$, the sensitivity is 1, as low as possible, but if $\mathbf{W}$ contains queries that are linear combinations of many cells, the value of $||\mathbf{WA}^{+}||_F$ can be very large and hence leads to high error. The other case is $\mathbf{A} = \mathbf{W}$, in which $\mathbf{A}$ and $\mathbf{W}$ are exactly the same. In such case, $||\mathbf{WA}^{+}||_F = \mathrm{rank}(\mathbf{W})$, which is small, but the strategy performs badly when the $||\mathbf{W}||$ is high. In many practical cases, the best strategy is one that achieves a good balance between the sensitivity and the similarity to $\mathbf{W}$. Optimization approaches to strategy selection are treated in detail in Sect. 7.

5.2 Total error equivalent workloads

Given the error analysis above, we now consider another form of equivalence among workloads: total error equivalence. Workloads that are total error equivalent are semantically different, but always have the same set of supporting strategies and the same total error under the matrix mechanism. On the other hand, semantically equivalent workloads are not necessarily error equivalent, although the minimum error of answering two semantically equivalent workloads is always the same, which will be addressed in Sect. 7.3.

Error equivalence is impossible among single queries. For two distinct queries $\mathbf{w}_1$ and $\mathbf{w}_2$, one can verify that the query strategy $\mathbf{A} = \begin{bmatrix} \mathbf{w}_1 \\ 2\mathbf{w}_2 \end{bmatrix}$ supports $\mathbf{w}_1$ and $\mathbf{w}_2$ and guarantees $\text{Error}_{\mathbf{A}}(\mathbf{w}_1) \neq \text{Error}_{\mathbf{A}}(\mathbf{w}_2)$. Therefore, there are no two queries that have the same error on all query strategies that support both of them. However, there exist pairs of query workloads whose total error is the same over all of their commonly supporting query strategies. These workloads are defined as total error equivalent.

Definition 12 Two query workloads $\mathbf{W}_1$ and $\mathbf{W}_2$ are called *total error equivalent*, if for any query strategy $\mathbf{A}$ that supports both $\mathbf{W}_1$ and $\mathbf{W}_2$, $\text{TotalError}_{\mathbf{A}}(\mathbf{W}_1) = \text{total error}\mathbf{A}\mathbf{W}_2$.

Analyzing Eq. (5) leads to the following condition of total error equivalent.

Proposition 10 *Given two query workloads* $\mathbf{W}_1$ *and* $\mathbf{W}_2$ *where* $\mathbf{W}_1$ *has at least as many queries as* $\mathbf{W}_1$. $\mathbf{W}_1$ *and* $\mathbf{W}_2$ *are total error equivalent, if and only if there exists an orthogonal matrix* $\mathbf{Q}$ *such that* $\mathbf{W}_1 = \mathbf{Q}\mathbf{W}_2$ *or* $\mathbf{W}_1 = \mathbf{Q}\begin{bmatrix} \mathbf{W}_2 \\ \mathbf{0} \end{bmatrix}$ *if* $\mathbf{W}_1$ *has more queries than* $\mathbf{W}_2$.

Proof ($\Leftarrow$) When $\mathbf{W}_1 = \mathbf{Q}\mathbf{W}_2$ or $\mathbf{W}_1 = \mathbf{Q}\begin{bmatrix} \mathbf{W}_2 \\ \mathbf{0} \end{bmatrix}$ if $\mathbf{W}_1$ has more queries than $\mathbf{W}_2$, we have $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2$. Notice that

$$\begin{aligned} ||\mathbf{W}\mathbf{A}^+||_F^2 &= \text{trace}\left(\mathbf{W}^T(\mathbf{A}^T\mathbf{A})^+\mathbf{W}\right) \\ &= \text{trace}\left(\mathbf{W}^T\mathbf{W}(\mathbf{A}^T\mathbf{A})^+\right), \end{aligned}$$

for any query strategy $\mathbf{A}$ that supports both $\mathbf{W}_1$ and $\mathbf{W}_2$, $\text{TotalError}_{\mathbf{A}}(\mathbf{W}_1) = \text{TotalError}_{\mathbf{A}}(\mathbf{W}_2)$.
($\Rightarrow$) If $\mathbf{W}_1^T\mathbf{W}_1 \neq \mathbf{W}_2^T\mathbf{W}_2$, consider the eigenvalue decomposition of $\mathbf{W}_1^T\mathbf{W}_1 - \mathbf{W}_2^T\mathbf{W}_2 = \mathbf{Q}\mathbf{D}\mathbf{Q}^T$ and $d_1, \ldots, d_n$ be the diagonal entries of $\mathbf{D}$. Without loss of generality, assume $d_1 \neq 0$ and let $\mathbf{D}' = \text{diag}(d_1', \ldots, d_n')$ where $d_1' = \sqrt{|d_1|}/\sqrt{|d_2| + \cdots + |d_n| + 1}$ and $d_2' = \cdots = d_n' = 1$. Let query strategy $\mathbf{A} = \mathbf{D}'\mathbf{Q}^T$. $\mathbf{A}$ supports $\mathbf{W}_1$ and $\mathbf{W}_2$ since it is full rank. Moreover,

$$\begin{aligned} &||\mathbf{W}_1\mathbf{A}^+||_F^2 - ||\mathbf{W}_2\mathbf{A}^+||_F^2 \\ &\quad = \text{trace}\left(\mathbf{W}_1^T(\mathbf{A}^T\mathbf{A})^+\mathbf{W}_1\right) - \text{trace}\left(\mathbf{W}_2^T(\mathbf{A}^T\mathbf{A})^+\mathbf{W}_2\right) \\ &\quad = \text{trace}\left(\mathbf{W}_1^T\mathbf{W}_1(\mathbf{A}^T\mathbf{A})^+\right) - \text{trace}\left(\mathbf{W}_2^T\mathbf{W}_2(\mathbf{A}^T\mathbf{A})^+\right) \\ &\quad = \text{trace}\left(\left(\mathbf{W}_1^T\mathbf{W}_1 - \mathbf{W}_2^T\mathbf{W}_2\right)(\mathbf{A}^T\mathbf{A})^+\right) \\ &\quad = \frac{d_1}{|d_1|}(|d_2| + \cdots + |d_n| + 1) + d_2 + \cdots + d_n \neq 0. \end{aligned}$$

When $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2$, there exist singular value decompositions $\mathbf{W}_1 = \mathbf{Q}_1\mathbf{D}_1\mathbf{P}^T$ and $\mathbf{W}_2 = \mathbf{Q}_2\mathbf{D}_2\mathbf{P}^T$, where the nonzero entries of $\mathbf{D}_1$ and $\mathbf{D}_2$ are the same. Thus, let $\mathbf{Q}_0 = \mathbf{Q}_1\mathbf{Q}_2^T$ or $\mathbf{Q}_0 = \mathbf{Q}_1\begin{bmatrix} \mathbf{Q}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$ if $\mathbf{W}_1$ has more queries than $\mathbf{W}_2$, and $\mathbf{W}_1 = \mathbf{Q}_0\mathbf{W}_2$. □

Noticing that any query strategy that supports $\mathbf{W}$ will support $\mathbf{Q}\mathbf{W}$ for any matrix $\mathbf{Q}$, the conclusion of Proposition 10 also indicates that the total error equivalent workloads share a same set of supporting strategies.

5.3 Equivalence between query strategies

Query strategies are essential to the matrix mechanism. When the matrix mechanism is instantiated with different strategies, it supports a different set of workloads and introduces different error. An important application of the error analysis is to determine sets of equivalent query strategies, which are entirely exchangeable under the matrix mechanism: Equivalent strategies support the same set of workloads and answer any query with exactly the same amount of error.

Definition 13 (*Query strategy equivalence*) Given two query strategies $\mathbf{A}_1$ and $\mathbf{A}_2$, we say that $\mathbf{A}_1$ and $\mathbf{A}_2$ are equivalent under $\mathcal{K}$ if they support the same sets of queries and for any query $\mathbf{w}$ that they support, $\text{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{w}) = \text{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{w})$.

To understand the equivalence between query strategies, let us review the error formulae in Eqs. (4) and (5). In both equations, the error is computed by $||\mathbf{A}||$ and a Frobenius norm term $||\mathbf{w}\mathbf{A}^+||_F^2$ and $||\mathbf{W}\mathbf{A}^+||_F^2$, respectively. According to the definition of the Frobenius norm,

$$\begin{aligned} ||\mathbf{w}\mathbf{A}^+||_F^2 &= \text{trace}\left(\mathbf{w}\mathbf{A}^+(\mathbf{w}\mathbf{A}^+)^T\right) \\ &= \text{trace}\left(\mathbf{w}(\mathbf{A}^T\mathbf{A})^+\mathbf{w}\right), \\ ||\mathbf{W}\mathbf{A}^+||_F^2 &= \text{trace}\left(\mathbf{W}\mathbf{A}^+(\mathbf{W}\mathbf{A}^+)^T\right) \\ &= \text{trace}\left(\mathbf{W}(\mathbf{A}^T\mathbf{A})^+\mathbf{W}^T\right). \end{aligned}$$

The right-hand sides of both equations above share a common term $(\mathbf{A}^T\mathbf{A})^+$, which we call an error profile.

Definition 14 Given a query strategy $\mathbf{A}$, the matrix $(\mathbf{A}^T\mathbf{A})^+$ is called the error profile of $\mathbf{A}$.

When the matrix mechanism is instantiated with $\mathbf{A}$, its error profile characterizes the distribution of the error of answering queries under the matrix mechanism: The diagonal entries contain the variance of error for each cell, and the off-diagonal entries encode the covariance of error between cells. Therefore, the error distribution of two query strategies is the same if their profiles differ by a constant factor, which is defined as profile equivalence.

$$\underset{\mathbf{H}'_4}{\begin{bmatrix}1&1&0&0\\0&0&1&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}} \qquad \underset{\mathbf{H}''_4}{\begin{bmatrix}1&1&0&0\\0&0&1&1\\\sqrt{2}&0&0&0\\0&\sqrt{2}&0&0\\0&0&\sqrt{2}&0\\0&0&0&\sqrt{2}\end{bmatrix}} \qquad \underset{\mathbf{Y}_4}{\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&0&0\\0&0&1&-1\end{bmatrix}}$$

Fig. 5 Profile-equivalent strategies using a cell list of size four

Definition 15 (*Profile equivalence between query strategies*) Given two query strategies $\mathbf{A}_1$ and $\mathbf{A}_2$, we say that $\mathbf{A}_1$ and $\mathbf{A}_2$ are profile equivalent if there exists a nonzero constant c such that $(\mathbf{A}_1^T\mathbf{A}_1)^+ = c(\mathbf{A}_2^T\mathbf{A}_2)^+$.

Profile-equivalent strategies are not always equivalent, since profile-equivalent strategies can have different error if their sensitivities do not differ by a factor of $1/c$. However, profile equivalence is also important since it is independent of the choice of $\mathcal{K}$, which is not true for query strategy equivalence. In addition, as it is proved later in this section, profile equivalence is a necessary condition to query strategy equivalence. Therefore, we first present some properties of profile equivalence and then study what condition leads to query strategy equivalence.

Example 6 Figure 5 contains three query strategies $\mathbf{H}'_4$, $\mathbf{H}''_4$, and $\mathbf{Y}_4$ that are profile equivalent. In particular, under the ϵ-differentially private matrix mechanism, $\mathbf{H}'_4$ is equivalent to $\mathbf{Y}_4$, but not equivalent to $\mathbf{H}''_4$, which has the same profile but smaller sensitivity. This also shows that $\mathbf{Y}$ is not optimal under ϵ-differential privacy because it is dominated by the improved strategy $\mathbf{H}''_4$.

There are other equivalent conditions to profile equivalence, based on strategy matrices and their transformations.

Proposition 11 *Given two query strategies* $\mathbf{A}_1$ *and* $\mathbf{A}_2$*, where* $\mathbf{A}_1$ *has at least as many rows as* $\mathbf{A}_2$*, all of the following conditions are equivalent*:

(i) $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *are profile equivalent;*
(ii) *There exists a nonzero constant c such that* $\mathbf{A}_1^T\mathbf{A}_1 = c\cdot\mathbf{A}_2^T\mathbf{A}_2$*;*
(iii) *There exists a nonzero constant c and an orthogonal matrix* $\mathbf{Q}$ *such that* $\mathbf{A}_1 = c\cdot\mathbf{Q}\mathbf{A}_2$ *or* $\mathbf{A}_1 = c\cdot\mathbf{Q}\begin{bmatrix}\mathbf{A}_2\\\mathbf{0}\end{bmatrix}$ *if* $\mathbf{A}_1$ *has more rows than* $\mathbf{A}_2$*;*

Proof (i) $\Leftrightarrow$ (ii) According to the definition of profile equivalence, $(\mathbf{A}_1^T\mathbf{A}_1)^+ = c\cdot(\mathbf{A}_2^T\mathbf{A}_2)^+$. Take the Moore–Penrose pseudoinverse to both sides of the equation and we have $\mathbf{A}_1^T\mathbf{A}_1 = \frac{1}{c}\cdot\mathbf{A}_2^T\mathbf{A}_2$.

(ii) $\Rightarrow$ (iii) Since $\mathbf{A}_1^T\mathbf{A}_1 = c\cdot\mathbf{A}_2^T\mathbf{A}_2$, there exist singular value decompositions of $\mathbf{A}_1 = \mathbf{Q}_1\mathbf{D}_1\mathbf{P}_1^T$ and $\mathbf{A}_2 = \mathbf{Q}_2\mathbf{D}_2\mathbf{P}_2^T$ such that $\mathbf{P}_1 = \mathbf{P}_2$ and the diagonal entries of $\mathbf{D}_1$ are equal to $\sqrt{c}$ times the diagonal entries of $\mathbf{D}_2$. If $\mathbf{A}_1$ has more rows than $\mathbf{A}_2$, the matrix $\mathbf{Q} = \mathbf{Q}_1\begin{bmatrix}\mathbf{Q}_2^T & \mathbf{0}\\\mathbf{0} & I\end{bmatrix}$ is the orthogonal matrix such that $\mathbf{A}_1 = \sqrt{c}\cdot\mathbf{Q}\begin{bmatrix}\mathbf{A}_2\\\mathbf{0}\end{bmatrix}$.

(iii) $\Rightarrow$ (ii) $\mathbf{A}_1^T\mathbf{A}_1 = c^2\cdot\mathbf{A}_2^T\mathbf{Q}^T\mathbf{Q}\mathbf{A}_2 = c^2\cdot\mathbf{A}_2^T\mathbf{A}_2$. $\square$

The conditions in Proposition 11 imply that profile-equivalent strategies support the same set of queries. In addition, for each query they support, the ratio between the error introduced by those strategies is the same.

Corollary 1 *Given two query strategies* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *that are profile equivalent, for any query* $\mathbf{W}$, $\mathbf{A}_1$ *supports* $\mathbf{W}$ *if and only if* $\mathbf{A}_2$ *supports* $\mathbf{W}$. *Furthermore, there exists a nonzero constant c such that given a differentially private algorithm* $\mathcal{K}$*, for any workload query* $\mathbf{W}$ *that* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *support,* $\text{TOTALERROR}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}) = c\cdot\text{TOTALERROR}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W})$.

The following proposition reveals that the strategy equivalence is a special case of profile equivalence with an extra constraint.

Proposition 12 *Two query strategies* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *are equivalent if they are profile equivalent and*

$$||\mathbf{A}_1||^2\left(\mathbf{A}_1^T\mathbf{A}_1\right)^+ = ||\mathbf{A}_2||^2\left(\mathbf{A}_2^T\mathbf{A}_2\right)^+,$$

In particular, $\mathbf{A}_1$ *and* $c\cdot\mathbf{A}_1$ *are equivalent for any nonzero scalar c.*

Proof ($\Leftarrow$) If $\mathbf{A}_1$ and $\mathbf{A}_2$ are profile equivalent, Corollary 1 indicates that they support the same set of query workloads. Furthermore, one can verify that for any workload query that $\mathbf{A}_1$ and $\mathbf{A}_2$ support,

$$\begin{aligned}||\mathbf{A}_1||^2||\mathbf{W}\mathbf{A}_1^+||_F^2 &= ||\mathbf{A}_1||^2\text{trace}\left(\mathbf{W}^T(\mathbf{A}_1^T\mathbf{A}_1)^+\mathbf{W}\right)\\ &= ||\mathbf{A}_2||^2\text{trace}\left(\mathbf{W}^T\left(\mathbf{A}_2^T\mathbf{A}_2\right)^+\mathbf{W}\right)\\ &= ||\mathbf{A}_2||^2||\mathbf{W}\mathbf{A}_2^+||_F^2.\end{aligned}$$

($\Rightarrow$) First, we prove that $\mathbf{A}_1^T\mathbf{A}_1$ and $\mathbf{A}_2^T\mathbf{A}_2$ have same eigenvectors. Otherwise, let $\mathbf{Q}_0$ be the matrix whose rows are orthogonal eigenvectors that are shared by $\mathbf{A}_1$ and $\mathbf{A}_2$, $\mathbf{Q}_1$ be the matrix whose rows are orthogonal eigenvectors of $\mathbf{A}_1$ that are supported by $\mathbf{A}_1$ and not eigenvectors of $\mathbf{A}_2$, and $\mathbf{Q}_2$ be the matrix whose rows are orthogonal eigenvectors of $\mathbf{A}_2$ that are supported by $\mathbf{A}_2$ and are not eigenvectors of $\mathbf{A}_1$. In addition, let $\mathbf{D}_1$ be the diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}_1^T\mathbf{A}_1$ corresponding to the rows of $\mathbf{Q}_1$ and let $\mathbf{D}_2$ be the diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}_2^T\mathbf{A}_2$ corresponding to the rows of $\mathbf{Q}_2$.

Noticing that the spanning space of $\mathbf{Q}_1$ contains all vectors that are supported by $\mathbf{A}_1$ and are orthogonal to all vectors in $\mathbf{Q}_0$ and so does the spanning space of $\mathbf{Q}_2$. Recall that the equivalent query strategies support the same set of queries; the rows in $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are actually two orthogonal bases to the same subspace. There hence exists an orthogonal matrix $\mathbf{Q}$ such that $\mathbf{Q}_1 = \mathbf{Q}\mathbf{Q}_2$. For any vector $\mathbf{v}$, $\mathbf{v}\mathbf{Q}_1$ is a query that $\mathbf{A}_1$ supports and

$$\begin{aligned}
&\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{v}\mathbf{Q}_1)\\
&\quad= P(\mathcal{K})||\mathbf{A}_1||^2||\mathbf{v}\mathbf{Q}_1\mathbf{A}_1^+||_F^2\\
&\quad= P(\mathcal{K})||\mathbf{A}_1||^2\text{trace}\left(\mathbf{v}\mathbf{Q}_1(\mathbf{A}_1^T\mathbf{A}_1)^+\mathbf{Q}_1^T\mathbf{v}^T\right)\\
&\quad= P(\mathcal{K})||\mathbf{A}_1||^2\mathbf{v}\mathbf{D}_1^{-1}\mathbf{v}^T,\\
&\textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{v}\mathbf{Q}_1)\\
&\quad= P(\mathcal{K})||\mathbf{A}_2||^2||\mathbf{v}\mathbf{Q}_1\mathbf{A}_2^+||_F^2\\
&\quad= P(\mathcal{K})||\mathbf{A}_2||^2\text{trace}\left(\mathbf{v}\mathbf{Q}\mathbf{Q}_2(\mathbf{A}_2^T\mathbf{A}_2)^+\mathbf{Q}_2^T\mathbf{Q}^T\mathbf{v}^T\right)\\
&\quad= P(\mathcal{K})||\mathbf{A}_2||^2\mathbf{v}\mathbf{Q}\mathbf{D}_2^{-1}\mathbf{Q}^T\mathbf{v}^T.
\end{aligned}$$

Since $\mathbf{A}_1$ and $\mathbf{A}_2$ are equivalent, $\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{v}\mathbf{Q}_1) = \textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{v}\mathbf{Q}_1)$ for any $\mathbf{v}$, which is equivalent to, for any $\mathbf{v}$,

$$\begin{aligned}
&\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{v}\mathbf{Q}_1) - \textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{v}\mathbf{Q}_1)\\
&\quad= P(\mathcal{K})\mathbf{v}\left(||\mathbf{A}_1||^2\mathbf{D}_1^{-1} - ||\mathbf{A}_2||^2\mathbf{Q}\mathbf{D}_2^{-1}\mathbf{Q}^T\right)\mathbf{v}^T\\
&\quad= 0.
\end{aligned}$$

Thus, $||\mathbf{A}_1||^2\mathbf{D}_1^{-1} = ||\mathbf{A}_2||^2\mathbf{Q}\mathbf{D}_2^{-1}\mathbf{Q}^T$, and we can consider $\mathbf{Q}(||\mathbf{A}_2||^2\mathbf{D}_2^{-1})\mathbf{Q}^T$ is an eigenvalue decomposition of matrix $||\mathbf{A}_1||^2\mathbf{D}_1^{-1}$. Recall $\mathbf{Q}_1 = \mathbf{Q}\mathbf{Q}_2$, and none of the rows of $\mathbf{Q}_1$ belongs to $\mathbf{Q}_2$. Therefore, there is no column in $\mathbf{Q}$ that consists one entry equal to 1 where all other entries are equal to 0, which indicates that all diagonal entries of $\mathbf{D}_2^{-1}$ should be equal. However, in such a case, the rows of $\mathbf{Q}_1$ will be eigenvectors of $\mathbf{A}_2^T\mathbf{A}_2$, which leads to a contradiction, and we know $\mathbf{A}_1^T\mathbf{A}_1$ and $\mathbf{A}_2^T\mathbf{A}_2$ must have same eigenvectors.

In addition, given an eigenvector $\mathbf{u}$ of $\mathbf{A}_1^T\mathbf{A}_1$ and $\mathbf{A}_2^T\mathbf{A}_2$ that $\mathbf{A}_1$ and $\mathbf{A}_2$ support, let $\mathbf{A}_1^T\mathbf{A}_1\mathbf{u}^T = \xi_1\mathbf{u}^T$ and $\mathbf{A}_2^T\mathbf{A}_2\mathbf{u}^T = \xi_2\mathbf{u}^T$. Since $\mathbf{A}_1$ and $\mathbf{A}_2$ support $\mathbf{u}$, $\xi_1 \neq 0$ and $\xi_2 \neq 0$. Furthermore,

$$\begin{aligned}
\textsc{Error}_{\mathbf{A}_1}(\mathbf{u}) &= P(\mathcal{K})||\mathbf{A}_1||^2||\mathbf{u}\mathbf{A}_1^+||_F^2\\
&= ||\mathbf{A}_1||^2\text{trace}\left(\mathbf{u}(\mathbf{A}_1^T\mathbf{A}_1)^+\mathbf{u}^T\right)\\
&= \frac{||\mathbf{A}_1||^2||\mathbf{u}||_2^2}{\xi_1},
\end{aligned}$$

$$\begin{aligned}
\textsc{Error}_{\mathbf{A}_2}(\mathbf{u}) &= P(\mathcal{K})||\mathbf{A}_2||^2||\mathbf{u}\mathbf{A}_2^+||_F^2\\
&= ||\mathbf{A}_2||^2\text{trace}\left(\mathbf{u}(\mathbf{A}_2^T\mathbf{A}_2)^+\mathbf{u}^T\right)\\
&= \frac{||\mathbf{A}_2||^2||\mathbf{u}||_2^2}{\xi_2},
\end{aligned}$$

Since $\textsc{Error}_{\mathbf{A}_1}(\mathbf{u}) = \textsc{Error}_{\mathbf{A}_2}(\mathbf{u})$, $||\mathbf{A}_1||^2\xi_2 = ||\mathbf{A}_2||^2\xi_1$. Noticing that it is true for all pairs of corresponding eigenvalues of $\mathbf{A}_1^T\mathbf{A}_1$ and $\mathbf{A}_2^T\mathbf{A}_2$, we have

$$||\mathbf{A}_2||^2\left(\mathbf{A}_1^T\mathbf{A}_1\right) = ||\mathbf{A}_1||^2\left(\mathbf{A}_2^T\mathbf{A}_2\right).$$

According to Corollary 1, $\mathbf{A}_1$ and $\mathbf{A}_2$ are profile equivalent and

$$||\mathbf{A}_1||^2\left(\mathbf{A}_1^T\mathbf{A}_1\right)^+ = ||\mathbf{A}_2||^2\left(\mathbf{A}_2^T\mathbf{A}_2\right)^+.$$

□

In particular, when $\mathcal{K}$ is under (ϵ, δ)-differential privacy, profile equivalence is equivalent to query strategy equivalence.

Proposition 13 *Given a differentially private algorithm $\mathcal{K}$, if $\mathcal{K}$ satisfies (ϵ, δ)-differential privacy, all profile-equivalent query strategies are equivalent.*

Proof When $\mathcal{K}$ is under (ϵ, δ)-differential privacy, sensitivity of $\mathbf{A}$ is computed by $||\mathbf{A}||_2$. Noticing that $||\mathbf{A}||_2^2$ is equal to the largest diagonal entry of matrix $\mathbf{A}^T\mathbf{A}$, given two query strategies $\mathbf{A}_1$ and $\mathbf{A}_2$ that are profile equivalent, by definition there exists a constant c such that $(\mathbf{A}_1^T\mathbf{A}_1)^+ = c\cdot(\mathbf{A}_2^T\mathbf{A}_2)^+$. Then, $c\cdot\mathbf{A}_1^T\mathbf{A}_1 = \mathbf{A}_2^T\mathbf{A}_2$ and $c\cdot||\mathbf{A}_1||_2^2 = ||\mathbf{A}_2||_2^2$. Substitute those values into Eq. (5), and we know $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}) = \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W})$ for any query workload $\mathbf{W}$ that $\mathbf{A}_1$ and $\mathbf{A}_2$ support. □

6 Application: analyzing $\mathbf{H}_n$ and $\mathbf{Y}_n$ using the matrix mechanism

In this section, we use our techniques to analyze and improve two previously proposed approaches for answering a specific workload: the set of all range queries over a (possibly multi-dimensional) domain [19,28]. We show that both approaches are instances of the matrix mechanism, each using a different query strategy designed to support the range query workload: In Xiao et al. [28], a wavelet transformation matrix is used as the strategy, and in Hay et al. [19], a hierarchical set of queries is used as the strategy. We will show that the seemingly distinct approaches have remarkably similar behavior: They have low (but not minimal) sensitivity, and they are highly accurate for range queries but much worse for queries that are not ranges. Although both techniques can support multi-dimensional range queries, we focus our analysis on one-dimensional range queries.

We briefly describe these techniques and how they can each be represented in matrix form. In the *hierarchical* scheme proposed in [19], the query strategy can be envisioned as a recursive partitioning of the domain. We consider the simple case of a binary partitioning, although higher branching factors were considered in [19]. The strategy includes the total sum over the whole domain, and then the count of each half of the domain, and so on, terminating with counts of individual elements of the domain. For a domain of size n (assumed for simplicity to be a power of 2), this results in a query strategy consisting of $2n-1$ rows. We represent this strategy as matrix $\mathbf{H}_n$, and $\mathbf{H}_4$ in Fig. 4 is a small instance of it.

In the *wavelet* scheme, proposed in [28], query strategies are based on the Haar wavelet. For one-dimensional range queries, the technique can also be envisioned as a hierarchical scheme, asking the total query, then asking for the difference between the left half and right half of the domain, continuing to recurse, asking for the difference in counts between each binary partition of the domain at each step. Though it is presented differently in [28], we prove later in this section the equivalence of that construction with our formulation $\mathbf{Y}_n$. This results in n queries—fewer than the hierarchical scheme of [19]. The matrix corresponding to this strategy is the matrix of the Haar wavelet transform, denoted $\mathbf{Y}_n$ in general, and illustrated as $\mathbf{Y}_4$ in Fig. 4. Thus, $\mathbf{H}_n$ is a rectangular $(2n-1)\times n$ query strategy and $\mathbf{Y}_n$ is an $n \times n$ query strategy.

As suggested by the examples in earlier sections, these seemingly different techniques have similar behavior. We analyze them in detail below, proving new bounds on the error for each technique, and proving new results about their relationship to one another. We also include $\mathbf{I}_n$ in the analysis, which is the strategy represented by the dimension n identity matrix, which asks for each individual count.

6.1 Representing the Haar wavelet technique

The representation of the wavelet approach in terms of strategy matrix $\mathbf{Y}_n$ is different from its original presentation in Xiao et al. [28]. The following theorem shows the equivalence of both representations.

Proposition 14 (Equivalence of Haar wavelet representations) *Let $\hat{\mathbf{x}}_{Haar}$ denote the estimate derived from the Haar wavelet approach of Xiao et al.* [28]. *Let $\hat{\mathbf{x}}_{\mathbf{Y}_n}$ denote the estimate from asking query $\mathbf{W}_n$. Then, $\hat{\mathbf{x}}_{Haar}$ and $\hat{\mathbf{x}}_{\mathbf{Y}_n}$ are equal in distribution, i.e., $Pr[\hat{\mathbf{x}}_{Haar} \le x] = Pr[\hat{\mathbf{x}}_{\mathbf{Y}_n} \le x]$ for any vector x.*

Proof Given vector $\mathbf{x}$, the Haar wavelet is defined in terms of a binary tree over $\mathbf{x}$ such that the leaves of the tree are $\mathbf{x}$.

Each node in the tree is associated with a coefficient. Coefficient c_i is defined as $c_i = (a_L - a_R)/2$ where a_L (a_R) is the average of the leaves in the left (right) subtree of c_i. Each c_i is associated with a weight $\mathcal{W}(c_i)$ which is equal to the number of leaves in subtree rooted at c_i (in addition, there is a coefficient c_0 that is equal to the average of $\mathbf{x}$ and $\mathcal{W}(c_0) = n$).

An equivalent definition for c_i is $c_i = \sum_{j=1}^{n} x_j z_i(j)$ where for $i > 0$,

$$z_i(j) = \begin{cases} 1/\mathcal{W}(c_i), & \text{if } j \text{ is in the left subtree of } c_i \\ -1/\mathcal{W}(c_i), & \text{if } j \text{ is in the right subtree of } c_i \\ 0, & \text{otherwise} \end{cases}$$

For $i = 0$, then $z_i(j)$ is equal to $1/\mathcal{W}(c_0)$ for all j.

Let $\mathbf{A}$ be a matrix where $a_{ij} = z_i(j)$. The ith row of $\mathbf{A}$ corresponds to coefficient c_i. Since there are n coefficients, $\mathbf{A}$ is an $n \times n$ matrix. The approach of [28] computes the following $\mathbf{y}_{\text{Haar}} = \mathbf{Ax} + \tilde{\mathbf{b}}_{\mathbf{A}}$ where $\tilde{\mathbf{b}}_{\mathbf{A}}$ is an $n \times 1$ vector whose ith entry is an independent sample from a Laplace distribution with scale $b_i = \frac{1+\log n}{\epsilon \mathcal{W}(c_i)}$. Observe that $\tilde{\mathbf{b}}_{\mathbf{A}}$ can be equivalently represented as:

$$\tilde{\mathbf{b}}_{\mathbf{A}} = \mathbf{R}^{-1}\left(\frac{1+\log n}{\epsilon}\right)\tilde{\mathbf{b}}$$

where $\mathbf{R}$ is an $n \times n$ diagonal matrix with $r_{ii} = \mathcal{W}(c_i)$. The estimate for $\mathbf{x}$ is then equal to:

$$\begin{aligned}\hat{\mathbf{x}}_{Haar} &= \mathbf{A}^{-1}\mathbf{y}_{Haar} = \mathbf{x} + \mathbf{A}^{-1}\tilde{\mathbf{b}}_{\mathbf{A}} \\ &= \mathbf{x} + \mathbf{A}^{-1}\mathbf{R}^{-1}\left(\frac{1+\log n}{\epsilon}\right)\tilde{\mathbf{b}} \\ &= \mathbf{x} + (\mathbf{RA})^{-1}\left(\frac{1+\log n}{\epsilon}\right)\tilde{\mathbf{b}}\end{aligned}$$

We now describe an equivalent approach based on the matrix $\mathbf{Y}_n$. Observe that $\mathbf{Y}_n = \mathbf{RA}$. The sensitivity of $\mathbf{Y}_n$ is $||\mathbf{Y}_n|| = 1 + \log n$. Using the matrix mechanism, the estimate $\hat{\mathbf{x}}_{\mathbf{Y}_n}$ is:

$$\begin{aligned}\hat{\mathbf{x}}_{\mathbf{Y}_n} &= \mathbf{Y}_n{}^{-1}\left(\mathbf{Y}_n\mathbf{x} + (\frac{||\mathbf{Y}_n||}{\epsilon})\tilde{\mathbf{b}})\right) \\ &= \mathbf{x} + \mathbf{Y}_n{}^{-1}\frac{||\mathbf{Y}_n||}{\epsilon}\tilde{\mathbf{b}} \\ &= \mathbf{x} + (\mathbf{RA})^{-1}\left(\frac{1+\log n}{\epsilon}\right)\tilde{\mathbf{b}}\end{aligned}$$

□

6.2 Similarity between $\mathbf{H}_n$ and $\mathbf{Y}_n$

Though represented differently, $\mathbf{H}_n$ and $\mathbf{Y}_n$ are actually very similar strategies under the matrix mechanism. In particular, a strategy matrix that is equivalent to $\mathbf{Y}_n$ can be achieved by removing the query of total sum and adding identity queries on each cell.

Theorem 1 *Let n be a power of 2, denoted as $n = 2^k$. Let $\mathbf{H}'_n$ be the matrix that results from removing the row of all 1s from matrix $\begin{bmatrix}\mathbf{H}_n\\ \mathbf{I}_n\end{bmatrix}$. Then, $\mathbf{H}'_n$ and $\mathbf{W}$ are equivalent strategies under both ϵ- and (ϵ, δ)-differential privacy.*

Proof Noticing that $\mathbf{H}'_n$ has the same L_1 and L_2 sensitivity with $\mathbf{Y}_n$, it is sufficient to prove $\mathbf{H}'^T_n\mathbf{H}'_n = \mathbf{Y}_n^T\mathbf{Y}_n$, which is equivalent to prove $\mathbf{H}_n^T\mathbf{H}_n = \mathbf{Y}_n^T\mathbf{Y}_n + \mathbf{1}_{n\times n} - \mathbf{I}_n$.

Recall $n = 2^k$, and we will prove the conclusion by induction on k. When $k = 1$,

$$\mathbf{H}_2^T\mathbf{H}_2 = \begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix},$$

$$\mathbf{Y}_2^T\mathbf{Y}_2 = \begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}.$$

Assume that the conclusion is correct for $k-1$. Since

$$\mathbf{H}_{2^k} = \begin{bmatrix}\mathbf{1}_{1\times 2^{k-1}} & \mathbf{1}_{1\times 2^{k-1}}\\ \mathbf{H}_{2^{k-1}} & 0\\ 0 & \mathbf{H}_{2^{k-1}}\end{bmatrix},$$

$$\mathbf{Y}_{2^k} = \begin{bmatrix}\mathbf{Y}_{2^{k-1}} & \begin{matrix}\mathbf{1}_{1\times 2^{k-1}}\\ 0\end{matrix}\\ \begin{matrix}-\mathbf{1}_{1\times 2^{k-1}}\\ 0\end{matrix} & \mathbf{Y}_{2^{k-1}}\end{bmatrix},$$

one can verify that $\mathbf{H}_{2^k}^T\mathbf{H}_{2^k} = \mathbf{Y}_{2^k}^T\mathbf{Y}_{2^k} + \mathbf{1}_{2^k\times 2^k} - \mathbf{I}_{2^k}$. □

Example 7 Figure 6 contains the strategy matrices $\mathbf{H}_4$, $\mathbf{H}'_4$, and $\mathbf{Y}_4$ and reveals the relationship between them. Adding $\mathbf{I}_4$ to $\mathbf{H}_4$ and removing the row of all 1s gives $\mathbf{H}'_4$. We find

$$\begin{bmatrix}1&1&1&1\\1&1&0&0\\0&0&1&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

(a) $\mathbf{H}_4$

$$\begin{bmatrix}1&1&0&0\\0&0&1&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

(b) $\mathbf{H}'_4$

$$\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&0&0\\0&0&1&-1\end{bmatrix}$$

(c) $\mathbf{Y}_4$

$$\begin{bmatrix}3&1&0&0\\1&3&0&0\\0&0&3&1\\0&0&1&3\end{bmatrix}$$

(d) $\mathbf{H}'^T_4\mathbf{H}'_4$, $\mathbf{Y}_4^T\mathbf{Y}_4$

Fig. 6 Strategy matrices $\mathbf{H}_4$, $\mathbf{H}'_4$, $\mathbf{Y}_4$, $\mathbf{H}'^T_4\mathbf{H}'_4$, and $\mathbf{Y}_4^T\mathbf{Y}_4$, which shows the similarity between $\mathbf{H}_4$ and $\mathbf{H}'_4$ as well as the equivalence between $\mathbf{H}'_4$ and $\mathbf{Y}_4$

that $\mathbf{H}'_4$ and $\mathbf{Y}_4$ are equivalent strategies (under both the ϵ- and (ϵ, δ)-differentially private matrix mechanisms) because their error profiles are equal (as indicated in Fig. 6d).

It follows from the similarity of $\mathbf{H}_n$ and $\mathbf{Y}_n$ that the error profiles are asymptotically equivalent to one another. We thus prove a close equivalence between the error of the two techniques:

Corollary 2 *For any linear counting query $\mathbf{w}$ and differentially private mechanism $\mathcal{K}$,*

$$\frac{1}{2}\textsc{Error}_{\mathcal{K},\mathbf{Y}}(\mathbf{w}) \le \textsc{Error}_{\mathcal{K},\mathbf{H}}(\mathbf{w}) \le 2\textsc{Error}_{\mathcal{K},\mathbf{Y}}(\mathbf{w}).$$

6.3 Error analysis for $\mathbf{I}_n$, $\mathbf{H}_n$, and $\mathbf{Y}_n$

In this part, we analyze the error for two specific workloads of interest. We focus on two typical workloads: $\mathbf{W}_R$, the set of all range queries, and $\mathbf{W}_{01}$, which includes arbitrary predicate queries, since it consists of all linear 0-1 queries. Note that attempting to use either of these workloads as a strategy leads to high sensitivity: The sensitivity of $\mathbf{W}_R$ is $O(n^2)$, while the sensitivity of $\mathbf{W}_{01}$ is $O(2^n)$. Here, we consider the total error as well as the maximum error under the matrix mechanism. The latter is defined as the worst-case error of a single query and denoted as MaxError.

In the original papers describing $\mathbf{H}_n$ and $\mathbf{Y}_n$ [19,28], both techniques are shown to have worst-case error, under ϵ-differential privacy, that is bounded by $O(\log^3 n)$ on $\mathbf{W}_R$. Both papers resort to experimental analysis to understand the distribution of total error across the class of range queries. We note that our results in Proposition 9 allow error for any query to be analyzed analytically.

Example 8 Figure 7 demonstrates the error of answering each range query in $\mathbf{W}_R$ under the Laplace mechanism with $n = 512$ and $\epsilon = 1$ using strategy matrices $\mathbf{H}_n$, $\mathbf{Y}_n$, and $\mathbf{I}_n$, respectively. Using the identity, $\mathbf{I}_n$ gives lower error on small ranges, but significant error on large ranges. Both $\mathbf{H}_n$ and $\mathbf{Y}_n$ smooth the error and have roughly similar patterns with $\mathbf{Y}_n$ favoring smaller ranges more than $\mathbf{H}_n$.

Next, we summarize the total error and the maximum error for these strategies. The following results tighten known bounds for $\mathbf{W}_R$ and show new bounds for $\mathbf{W}_{01}$ with the ϵ-differential privacy.

Theorem 2 (Error and Maximum Error) *The total error and the maximum error on workloads $\mathbf{W}_R$ and $\mathbf{W}_{01}$ using strategies $\mathbf{H}_n$, $\mathbf{Y}_n$, and $\mathbf{I}_n$ under the ϵ- and (ϵ, δ)-matrix mechanism is given in Table 1.*

Proof Here, we focus on the case of ϵ-matrix mechanism since the proof on (ϵ, δ)-matrix mechanism is exactly the same.

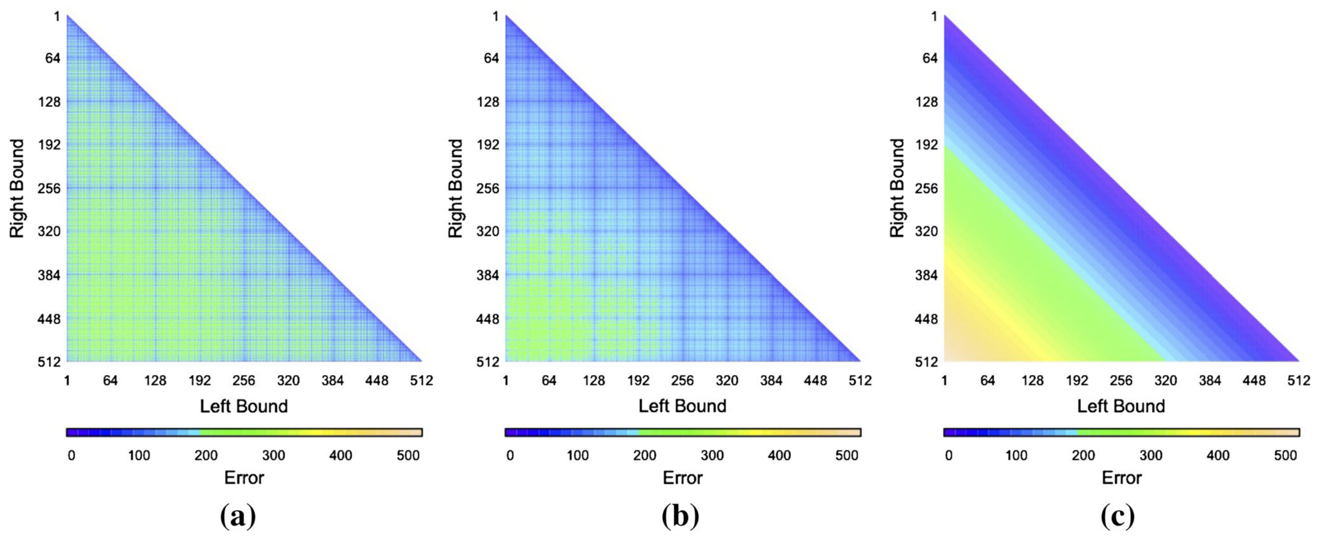

Fig. 7 Error of each range query in $\mathbf{W}_R$ using different strategies under ϵ-matrix mechanism with $n = 512$, $\epsilon = 1$. **a** $\mathbf{H}_n$ as the strategy matrix. **b** $\mathbf{Y}_n$ as the strategy matrix. **c** $\mathbf{I}_n$ as the strategy matrix

Table 1 Total and maximum error of $\mathbf{H}_n$, $\mathbf{Y}_n$, and $\mathbf{I}_n$ on all predicate queries

	$\mathbf{H}_n$	$\mathbf{Y}_n$	$\mathbf{I}_n$
(a) Under ϵ-differential privacy			
TotalError			
$\mathbf{W}_R$	$\Theta(n^2 \log^3 n/\epsilon^2)$	$\Theta(n^2 \log^3 n/\epsilon^2)$	$\Theta(n^3/\epsilon^2)$
$\mathbf{W}_{01}$	$\Theta(n2^n \log^2 n/\epsilon^2)$	$\Theta(n2^n \log^2 n/\epsilon^2)$	$\Theta(n2^n/\epsilon^2)$
MaxError			
$\mathbf{W}_R$	$\Theta(\log^3 n/\epsilon^2)$	$\Theta(\log^3 n/\epsilon^2)$	$\Theta(n/\epsilon^2)$
$\mathbf{W}_{01}$	$\Theta(n \log^2 n/\epsilon^2)$	$\Theta(n \log^2 n/\epsilon^2)$	$\Theta(n/\epsilon^2)$
(b) Under (ϵ, δ)-differential privacy			
TotalError			
$\mathbf{W}_R$	$\Theta(n^2 \log^2 n \log(1/\delta)/\epsilon^2)$	$\Theta(n^2 \log^2 n \log(1/\delta)/\epsilon^2)$	$\Theta(n^3 \log(1/\delta)/\epsilon^2)$
$\mathbf{W}_{01}$	$\Theta(n2^n \log n \log(1/\delta)/\epsilon^2)$	$\Theta(n2^n \log n \log(1/\delta)/\epsilon^2)$	$\Theta(n2^n \log(1/\delta)/\epsilon^2)$
MaxError			
$\mathbf{W}_R$	$\Theta(\log^2 n \log(1/\delta)/\epsilon^2)$	$\Theta(\log^2 n \log(1/\delta)/\epsilon^2)$	$\Theta(n \log(1/\delta)/\epsilon^2)$
$\mathbf{W}_{01}$	$\Theta(n \log n \log(1/\delta)/\epsilon^2)$	$\Theta(n \log n \log(1/\delta)/\epsilon^2)$	$\Theta(n \log(1/\delta)/\epsilon^2)$

Since $\mathbf{W}_n$ and $\mathbf{H}_n$ are asymptotically equivalent, we can derive the error bounds for either. We analyze the error of $\mathbf{W}_n$. Let $n = 2^{k+1}$, consider the range query $[2^k - \frac{1}{3}(4^{\lfloor\frac{k-1}{2}\rfloor+1} - 1), 2^k + \frac{1}{3}(4^{\lfloor\frac{k-1}{2}\rfloor+1} - 1)]$. The error of this query is $\Theta(\log^3 n)$, which follows from algebraic manipulation of Eq. 5, facilitated by knowing the eigendecomposition of $(\mathbf{W}_n^T\mathbf{W}_n)^+$. Since Xiao et al. [28] have already shown that the worst-case error of $\mathbf{W}_n$ is $O(\log^3 n)$, we know the maximum error of answering any query in $\mathbf{W}_R$ is $\Theta(\log^3 n)$.

Moreover, it follows from algebraic manipulation that the error of answering any query $\mathbf{w}$ where the number of nonzero entries is 1 is $O(\log^2 n)$. Therefore, the error of any 0-1 query is $O(n \log^2 n)$. Consider the query $(0, 1, 0, 1, \ldots, 0, 1)$: It can be shown to have error $\Theta(n \log^2 n)$. Therefore, the maximum error of answering any query in $\mathbf{W}_{01}$ is $\Theta(n \log^2 n)$.

Recall that when $\mathcal{K}$ is the Laplace mechanism,

$$\text{TotalError}_{\mathbf{A}}(\mathbf{W}) = \frac{2}{\epsilon^2}||\mathbf{A}||_1^2||\mathbf{WA}^+||_F^2.$$

Total error of workloads $\mathbf{W}_R$, $\mathbf{W}_{01}$ can be computed by applying the equation above to strategies $\mathbf{H}_n$, $\mathbf{W}_n$ and $\mathbf{I}_n$. □

7 Optimal strategy selection

An essential task in deploying the matrix mechanism is selecting an appropriate query strategy $\mathbf{A}$ for a given query workload $\mathbf{W}$. In this section, we present techniques that generate optimal or approximately optimal strategies for a given workload under the matrix mechanism, as well as a heuristic that can enhance any given strategy. We first define our main problem as follows:

Problem 1 (MINERROR) Given a query workload $\mathbf{W}$ and a differentially private algorithm $\mathcal{K}$, find a query strategy $\mathbf{A}$ that supports $\mathbf{W}$ and minimizes $\text{TOTALERROR}_{\mathcal{K},\mathbf{A}}(\mathbf{W})$.

Since both $\mathbf{W}$ and $-\mathbf{W}$ support $\mathbf{W}$ but $\mathbf{0} = \mathbf{W} + (-\mathbf{W})$ does not, the MINERROR problem is non-convex. In this section, we will formulate the MINERROR problem as a semidefinite program with rank constraints, which is a non-convex variant of a semidefinite program. We then discuss the problem in two cases corresponding to the differential privacy guarantee of $\mathcal{K}$. A general technique that can be used to improve a query strategy is also provided in the later section. We also show that two semantically equivalent workloads yield the same minimum total error at the end of this section.

7.1 Formulating the MINERROR problem

Here, we show that the MINERROR problem under ϵ-differential privacy can be expressed as a semidefinite program with rank constraints. While rank constraints make the semidefinite program non-convex, there are algorithms that can solve such problems by iteratively solving a pair of related semidefinite programs.

Theorem 3 *Given an $m \times n$ workload $\mathbf{W}$, MEDP(Program 7.1) is a semidefinite program with rank constraint whose solution is the tuple $(\mathbf{A}, \mathbf{u}, \mathbf{X})$ and the $m' \times n$ strategy $\mathbf{A}$ minimizes $\text{TOTALERROR}_{\mathcal{K},\mathbf{A}}(\mathbf{W})$ among all $m' \times n$ strategies.*

Program 7.1 (MEDP): minimizing the total error with $\mathcal{K}$ under ϵ-differential privacy

Given: $\mathbf{W} \in \mathbb{R}^{m\times n}$
Minimize: $u_1 + u_2 + \cdots + u_m$
Subject to: For $i \in [m]$: $\mathbf{w}_i$ is the i-th row of $\mathbf{W}$.

$$\begin{bmatrix} \mathbf{X} & \mathbf{w}_i^T \\ \mathbf{w}_i & u_i \end{bmatrix} \succeq 0 \tag{6}$$

$$||\mathbf{A}||_1 \leq 1 \tag{7}$$

$$\text{rank}\left(\begin{bmatrix} \mathbf{I}_{m'} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{X} \end{bmatrix}\right) = m' \tag{8}$$

Proof To prove that the output strategy $\mathbf{A}$ of MEDP is an optimal $m' \times n$ strategy to the MINERROR problem, one needs to show that the solution of MEDP supports $\mathbf{W}$ and the optimization goal of MEDP is equivalent with the MINERROR problem.

The semidefinite condition in (6) is important, which guarantees that there exists a matrix $\mathbf{A}'$ such that $\mathbf{A}'^T\mathbf{A}' = \mathbf{X}$, $\mathbf{A}'$ supports $\mathbf{w}_i$, and $u_i \geq ||\mathbf{w}_i\mathbf{A}'^+||_F^2$. According to the properties of positive semidefinite matrices, it is a symmetric matrix with nonnegative eigenvalues. Let $\mathbf{X} = \mathbf{P}\mathbf{\Sigma}\mathbf{P}^T$ be an eigenvalue decomposition of matrix $\mathbf{X}$. Consider the matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{P}^T & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} & \mathbf{w}_i^T \\ \mathbf{w}_i & u_i \end{bmatrix} \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{\Sigma} & (\mathbf{w}_i\mathbf{P})^T \\ \mathbf{w}_i\mathbf{P} & u_i \end{bmatrix},$$

(6) holds if and only if $\mathbf{Y} \succeq 0$. Since $\mathbf{\Sigma}$ is a diagonal matrix, if its jth diagonal entry is 0, the jth entry of $\mathbf{w}_i\mathbf{P}$ must be 0 as well. Otherwise, $\mathbf{Y}$ cannot be positive semidefinite no matter the value of u_i. Recall that the diagonal entries of $\mathbf{\Sigma}$ are eigenvalues of $\mathbf{X}$ and hence are nonnegative. Let $\mathbf{D}$ be the diagonal matrix whose diagonal entries are the square roots of diagonal entries of $\mathbf{\Sigma}$. We know $\mathbf{D}$ supports $\mathbf{w}_i\mathbf{P}$. Then, $\mathbf{A}' = \mathbf{D}\mathbf{P}^T$ supports $\mathbf{w}_i$ and $\mathbf{A}'^T\mathbf{A}' = \mathbf{X}$. In addition, let $\mathbf{Y}'$ be the matrix that is constructed by removing all 0 columns and rows from $\mathbf{Y}$. For any $\mathbf{w}_i$ that is supported by $\mathbf{A}'$, $\mathbf{Y} \succeq 0$ is equivalent to $|\mathbf{Y}'| \geq 0$. The expansion of $|\mathbf{Y}'|$ implies that the determinant is nonnegative if and only if $u_i \geq ||\mathbf{w}_i\mathbf{A}'^+||_F^2$. Since the goal of the optimization problem is to minimize the sum of u_i, when the optimal case is achieved, we must have $u_i = ||\mathbf{w}_i\mathbf{A}'^+||_F^2 = \mathbf{w}_i(\mathbf{A}'^T\mathbf{A}')^+\mathbf{w}_i^T = \mathbf{w}_i\mathbf{X}^+\mathbf{w}_i^T$. Furthermore, (8) guarantees that $\mathbf{X} = \mathbf{A}^T\mathbf{A}$, and hence $u_i = ||\mathbf{w}_i\mathbf{A}^+||_F^2$.

According to Proposition 12, applying a nonzero scalar c to a query strategy $\mathbf{A}$ leads to its equivalent strategy. Therefore, the condition (7) does not limit the scope of query strategies to be considered since any query strategy has equivalent strategies with sensitivity no more than 1. This constraint is as well a convex constraint since the sensitivity of $\mathbf{A}$ is convex for both ϵ- and (ϵ, δ)-differential privacy. Noticing $||\mathbf{w}_i(c \cdot \mathbf{A})^+||_F = ||\mathbf{w}_i\mathbf{A}^+||_F/c < ||\mathbf{w}_i\mathbf{A}^+||_F$ for any $c > 1$, $||\mathbf{A}||_1$ must be 1 in the optimal case.

Above all, any solution to MEDP supports $\mathbf{W}$. When the optimal case is achieved, $||\mathbf{A}||_1 = 1$ and $u_i = ||\mathbf{w}_i\mathbf{A}^+||_F^2 = ||\mathbf{A}||_1^2||\mathbf{w}_i\mathbf{A}^+||_F^2 = P(\mathcal{K})\text{ERROR}_{\mathcal{K},\mathbf{A}}(\mathbf{w}_i)$. Therefore, the goal of the optimization, minimizing $\sum_{i=1}^m u_i$, is equivalent to minimizing

$$\sum\nolimits_{i=1}^{m} \text{ERROR}_{\mathcal{K},\mathbf{A}}(\mathbf{w}_i) = \text{TOTALERROR}_{\mathcal{K},\mathbf{A}}(\mathbf{W}).$$

□

Theorem 3 provides the best strategy to the MINERROR problem with at most m' queries. If the optimal strategy has $m'' < m'$ queries, then MEDP will return an $m' \times n$ matrix with $m' - m''$ rows of 0s. In addition, if the workload only

Program 7.2 (**MEADP**): Minimizing the Total Error with $\mathcal{K}$ under (ϵ, δ)-differential privacy

Given: $\mathbf{W} \in \mathbb{R}^{m\times n}$.

Minimize: $u_1 + u_2 + \cdots + u_m$.

Subject to: For $i \in [m]$: $\mathbf{w}_i$ is the i-th row of $\mathbf{W}$.

$$\begin{bmatrix} \mathbf{X} & \mathbf{w}_i^T \\ \mathbf{w}_i & u_i \end{bmatrix} \succeq 0$$

$$\mathbf{X}_{ii} \leq 1, \quad i \in [n].$$

contains queries with coefficients in $\{-1, 0, 1\}$, we can show that n^2 is an upper bound on the number of queries in the optimal strategy [20].

In addition, since MEDP encodes the error of each query $\mathbf{w}_i$ in query workload $\mathbf{W}$, we can actually use another convex function of $u_1, \ldots, u_m$ to take the place of $u_1 + \cdots + u_m$ in the optimization goal. One variation to the optimization goal is $\max_i u_i$, under which the result of MEDP becomes the query strategy that minimizes the maximum error of all queries in $\mathbf{W}$.

Dattorro [5] shows that solving a semidefinite program with rank constraints can be converted into solving two semidefinite programs iteratively. The convergence follows the widely used trace heuristic for rank minimization. We are not aware of results that quantify the number of iterations that are required for convergence. However, notice that it takes $O(n^4)$ time to solve a semidefinite program with an $n \times n$ semidefinite constraint matrix and in MEDP; there are m semidefinite constraint matrices with size $m+n$, which can be represented as a semidefinite constraint matrix with size $m(m+n)$. Thus, the complexity of solving our semidefinite program with rank constraints is at least $O(m^4(m+n)^4)$.

The difficulty of MEDP comes from the rank constraint (8), which is used to connect $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}$ since we need $\mathbf{A}^T\mathbf{A}$ to compute $||\mathbf{w}_i\mathbf{A}^+||_F^2$ and $\mathbf{A}$ to compute $||\mathbf{A}||_1$. However, when $\mathcal{K}$ is based on (ϵ, δ)-differential privacy, $||\mathbf{A}||_2$ can be computed directly from $\mathbf{A}^T\mathbf{A}$. In that case, $\mathbf{A}$ is not necessary in the optimization problem and the rank constraint can be removed. The optimization problem can then be reduced to a semidefinite program, which can be solved more efficiently.

Theorem 4 *Given an $m \times n$ workload $\mathbf{W}$, MEADP(Program 7.2) is a semidefinite program whose solution is the tuple $(\mathbf{X}, \mathbf{u})$ and any $m' \times n$ strategy $\mathbf{A}$ such that $\mathbf{X} = \mathbf{A}^T\mathbf{A}$ minimizes* $\text{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W})$ *among all strategies under (ϵ, δ)-differential privacy.*

7.1.1 Minimizing the sensitivity under ϵ-differential privacy

Similar to the MinError problem, another important task is to find the best strategy with a desired error distribution for the workload queries. In the language of the matrix mechanism, this means finding the best strategy among all profile-equivalent strategies:

Problem 2 Given an error profile $\mathbf{M}$, find the query matrix $\mathbf{A}$ whose error profile is $\mathbf{M}$ and has the minimum sensitivity under ϵ-differential privacy.

Program 7.3 Minimizing the sensitivity under ϵ-differential privacy

Given: $\mathbf{M} \in \mathbb{R}^{n\times n}$.

Minimize: r.

Subject to: $||\mathbf{A}||_1 \leq r$;

$$\text{rank}\left(\begin{bmatrix} \mathbf{I}_n & \mathbf{A} \\ \mathbf{A}^T & \mathbf{M}^+ \end{bmatrix}\right) = n.$$

Unfortunately, similar to the MinError problem, Problem 2 is non-convex. Problem 2 can also be formulated as a semidefinite program with rank constraint, as stated below.

Theorem 5 *Given an error profile $\mathbf{M}$, Program 7.3 is a semidefinite program with rank constraint that outputs an $m \times n$ matrix $\mathbf{A}$ such that $(\mathbf{A}^T\mathbf{A})^+ = \mathbf{M}$ with $||\mathbf{A}||_1$ minimized.*

7.2 Augmentation heuristic

We formalize below the following intuition that applies to the matrix mechanism: As far as the error profile is concerned, additional noisy query answers can never detract from query accuracy as they must have some information content useful to one or more queries. Therefore, if $\mathbf{A}'$ is a query strategy formed by augmenting the query strategy $\mathbf{A}$ with additional rows, $||\mathbf{W}\mathbf{A}'^+||_F \leq ||\mathbf{W}\mathbf{A}^+||_F$.

Theorem 6 (Augmenting a strategy) *Let $\mathbf{A}$ be a query strategy, and consider a new strategy $\mathbf{A}'$ obtained from $\mathbf{A}$ by adding the additional rows of strategy $\mathbf{B}$, so that $\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$. For any workload $\mathbf{W}$ that $\mathbf{A}$ supports, $\mathbf{A}'$ supports $\mathbf{W}$ and*

$$||\mathbf{W}\mathbf{A}'^+||_F \leq ||\mathbf{W}\mathbf{A}^+||_F.$$

Further, $||\mathbf{W}\mathbf{A}'^+||_F = ||\mathbf{W}\mathbf{A}^+||_F$ if and only if $\mathbf{W}\mathbf{A}'^+ = \begin{bmatrix} \mathbf{W}\mathbf{A}^+ \\ \mathbf{0} \end{bmatrix}$.

Proof Since $\mathbf{A}$ supports $\mathbf{W}$, $\mathbf{A}'$ supports $\mathbf{W}$ as well. Noticing padding $\mathbf{W}\mathbf{A}^+$ with some 0s gives a solution to equation $\mathbf{X}\mathbf{A}' = \mathbf{W}$, according to Proposition 17, $||\mathbf{W}\mathbf{A}'^+||_F \leq ||\mathbf{W}\mathbf{A}^+||_F$.

Let $\mathbf{w}_1, \ldots, \mathbf{w}_m$ be rows of $\mathbf{W}$. Noticing that

$$||\mathbf{WA}^+||_F = \sum_{i=1}^{m} ||\mathbf{w}_i \mathbf{A}^+||_F;$$
$$||\mathbf{WA}'^+||_F = \sum_{i=1}^{m} ||\mathbf{w}_i \mathbf{A}'^+||_F;$$
$$||\mathbf{w}_i \mathbf{A}'^+||_F \le ||\mathbf{w}_i \mathbf{A}^+||_F, \quad i = 1, \ldots, m.$$

Therefore, $||\mathbf{WA}'^+||_F = ||\mathbf{WA}^+||_F$ if and only if $||\mathbf{w}_i\mathbf{A}'^+||_F = ||\mathbf{w}_i\mathbf{A}^+||_F$ for all $i = 1, \ldots, m$. Thus, it is sufficient to consider the condition that $||\mathbf{wA}'^+||_F = ||\mathbf{wA}^+||_F$ for a single query $\mathbf{w}$ that $\mathbf{A}$ supports.

Given two distinct solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ to equation $\mathbf{xA}' = \mathbf{w}$. If $||\mathbf{x}_1||_F = ||\mathbf{x}_2||_F$, noticing the Frobenius norm is convex, we have $||(\mathbf{x}_1 + \mathbf{x}_2)/2||_F < ||\mathbf{x}_1||_F = ||\mathbf{x}_2||_F$. Since $(\mathbf{x}_1 + \mathbf{x}_2)/2$ is also a solution to equation $\mathbf{xA}' = \mathbf{w}$, the solution of $\mathbf{xA}' = \mathbf{w}$ with minimized Frobenius norm is unique and $||\mathbf{wA}'^+||_F = ||\mathbf{wA}^+||_F$ if and only if $\mathbf{wA}'^+$ is equal to $\mathbf{wA}^+$ padding with 0s. □

This improvement in the error profile may have a cost—namely, augmenting $\mathbf{A}$ with strategy $\mathbf{B}$ may lead to a strategy $\mathbf{A}'$ with greater sensitivity than $\mathbf{A}$. A heuristic that follows from Theorem 6 is to augment strategy $\mathbf{A}$ only by completing deficient columns, that is by adding rows with nonzero entries only in columns whose absolute column sums are less than the sensitivity of $\mathbf{A}$. In this case, the augmentation does not increase sensitivity and is guaranteed to strictly improve accuracy for any query with a nonzero coefficient in an augmented column.

Our techniques could also be used to reason formally about augmentations that do incur a sensitivity cost. We leave this as future work, as it is relevant primarily to an interactive differentially private mechanism which is not our focus here.

7.3 The MinError problem for semantically equivalent workloads

As defined in Definition 5, semantically equivalent workloads only differ in their representations, and answering them should introduce exactly the same amount of error. However, as pointed out in Sect. 5.2, semantically equivalent workloads may not be error equivalent. On the other hand, semantically equivalent workloads do have the same minimum error, shown as follows.

Theorem 7 *Given workload* $\mathbf{W}_1$ *over cell conditions* Φ_1 *and workload* $\mathbf{W}_2$ *over cell conditions* Φ_2 *such that* $(\mathbf{W}_1, \Phi_1) \equiv (\mathbf{W}_2, \Phi_2)$, $\min_{\mathbf{A}} \text{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}_1) = \min_{\mathbf{A}} \text{TOTALERROR}\mathcal{K}, \mathbf{AW}_2$ *for any differentially private algorithm* $\mathcal{K}$.

Proof By symmetry, it is sufficient to prove that for any strategy $\mathbf{A}_1$ that supports $\mathbf{W}_1$, there exists a strategy $\mathbf{A}_2$ such that $\mathbf{A}_2$ supports $\mathbf{W}_2$ and

$$\text{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) \ge \text{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2).$$

As described in Proposition 1, there are five operations to generate semantic equivalent workloads. We prove that such an $\mathbf{A}_2$ exists for each of the five operations.

If $\mathbf{W}_2$ can be formed by permuting rows and columns of $\mathbf{W}_1$, there exist permutation matrices $\mathbf{P}$ and $\mathbf{Q}$ such that $\mathbf{W}_2 = \mathbf{PW}_1\mathbf{Q}$. Then, $\mathbf{A}_1\mathbf{Q}$ supports $\mathbf{W}_2$ and $\text{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) = \text{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$.

If $\mathbf{W}_2$ results from splitting one of the columns of $\mathbf{W}_1$, without loss of generality, we assume that $\mathbf{W}_2$ is generated by splitting the last column of $\mathbf{W}_1$ into two columns. Let $\mathbf{A}_2$ be the matrix that is generated by applying the same split to the strategy matrix $\mathbf{A}_1$. Then, it is clear that $\mathbf{W}_1\mathbf{A}_1^+\mathbf{A}_2 = \mathbf{W}_2$. Therefore, $\mathbf{A}_2$ supports $\mathbf{W}_2$ and $\text{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) = \text{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$.

If $\mathbf{W}_2$ results from combining two of the columns of $\mathbf{W}_1$ with the same entries, we assume that the last two columns of $\mathbf{W}_1$ have the same entries and removing one of them gives us $\mathbf{W}_2$. Let $\mathbf{A}_2$ be the matrix that is generated by removing the last column of $\mathbf{A}_2$. Then, $\mathbf{W}_1\mathbf{A}_1^+\mathbf{A}_2 = \mathbf{W}_2$ and $\mathbf{A}_2$ hence supports $\mathbf{W}_2$. Noticing that $||\mathbf{A}_2|| \le ||\mathbf{A}_1||$ for any $\mathcal{K}$, $\text{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) \ge \text{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$.

If $\mathbf{W}_2$ results from adding columns of 0s to $\mathbf{W}_1$, let $\mathbf{A}_2$ be the matrix that is generated by adding corresponding columns of 0s to $\mathbf{A}_1$. Then, $\mathbf{W}_1\mathbf{A}_1^+\mathbf{A}_2 = \mathbf{W}_2$. Thus, $\mathbf{A}_2$ supports $\mathbf{W}_2$ and $\text{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) = \text{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$.

If $\mathbf{W}_2$ results from removing columns of 0s to $\mathbf{W}_1$, let $\mathbf{A}_2$ be the matrix that is generated by removing corresponding columns of 0s to $\mathbf{A}_1$. Then, $\mathbf{W}_1\mathbf{A}_1^+\mathbf{A}_2 = \mathbf{W}_2$ and $\mathbf{A}_2$ hence supports $\mathbf{W}_2$. Noticing that $||\mathbf{A}_2|| \le ||\mathbf{A}_1||$ for any $\mathcal{K}$,

$$\text{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) \ge \text{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$$

for any $\mathcal{K}$. □

Since we do not consider cell conditions in the proof of Theorem 7, two workloads $\mathbf{W}_1$ and $\mathbf{W}_2$ have the same minimum total error if they can be converted to one another with the matrix operations mentioned in Proposition 1. On the other hand, according to Theorem 7, given a workload $\mathbf{W}$ and its corresponding list of cell conditions Φ, we can simplify $\mathbf{W}$ and Φ by:

– Removing all 0 columns in $\mathbf{W}$ and discarding their corresponding cell conditions from Φ.
– Merging all columns with the same entries and their corresponding cell conditions.

Table 2 Special cases of the matrix mechanism presented in the literature

Workload class	References	Privacy guarantee
1-Dimensional range queries	[19,26,28,31]	ϵ-Differential privacy
Low-order marginal queries	[2]	ϵ-Differential privacy
Datacube	[6]	ϵ-Differential privacy
Low-rank workloads	[32]	ϵ-Differential privacy
Any workloads	[21,24]	(ϵ, δ)-Differential privacy

In particular, if $\mathbf{W}$ is a subset of all range queries, $\mathbf{W}_R$, we can always simplify $\mathbf{W}$ and its cell conditions such that the number of cell conditions is no more than twice the number of queries in $\mathbf{W}$.

Corollary 3 (Cell condition simplification) *Given a workload* $\mathbf{W}_1 \subseteq \mathbf{W}_R$ *with* m *queries and its corresponding list of cell conditions,* Φ_1*, there exists a workload* $\mathbf{W}_2$ *and a list of cell conditions,* Φ_2*, such that* $(\mathbf{W}_1, \Phi_1) \equiv (\mathbf{W}_2, \Phi_2)$, $|\Phi_2| \leq 2m - 1$.

7.4 Strategy selection in practice

As we have shown above, the strategy selection problem is difficult to solve in general. Specific strategies, or efficient strategy selection methods, have been developed that perform well for the special case of workloads derived from a given class of workloads (e.g., range queries or sets of data cubes). Table 2 summarizes these results and provides references to the relevant literature.

Although they will not, in general, offer the best possible error rates achievable using the matrix mechanism, these methods still offer reduced error over baseline methods. We present experimental results comparing some of these methods with the baseline methods (Laplace or Gaussian) on different workloads. Below, when we mention a class of queries (e.g., 1-dimensional range queries or 2-way marginal queries), our results concern the workload consisting of *all* queries in the class.

- For the workload of all 1-dimensional range queries under ϵ-differential privacy, we compare, in Fig. 8a, the Laplace mechanism with instances of the matrix mechanism using the hierarchical strategy [19], the wavelet strategy [28], and the hierarchical strategy with optimal branching factor [26] (denoted as Optimal Hierarchical). We vary the number of cells from 512 to 2048.
- For workloads of all 1- or multi-dimensional range queries under (ϵ, δ)-differential privacy, we compare, in Fig. 8b, the Gaussian mechanism with the matrix mechanism using the hierarchical strategy [19], the wavelet strategy [28], and the strategy generated using the eigen-design algorithm [21]. We consider domains with 2048 cells but different numbers of dimensions, varying from a 1-dimensional domain with 2048 cells to a 4-dimensional domain with 8, 8, 8, and 4 cells on each dimension (denoted as $[8 \cdot 8 \cdot 8 \cdot 4]$).
- For workloads of all 1- and 2-way marginal queries under (ϵ, δ)-differential privacy, we compare, in Fig. 8c, the Gaussian mechanism with the matrix mechanism using the Fourier strategy [2], the strategy generated by differentially private data cubes [6] (denoted as DataCube), and the strategy generated using the eigen-design algorithm [21]. We test on domains with 2048 cells but with different numbers of dimensions, varying from a 3-dimensional space with 16, 16, 8 cells on each dimension to an 11-dimensional space with 2 cells on each dimension (denoted as $[2^{11}]$).

Since the error rate of the matrix mechanism only depends on the domain size and workload and is independent of the underlying dataset, we only show shapes of the cell list and omit any concrete dataset. Further, since changing the privacy parameters impacts both the baseline mechanism and instances of the matrix mechanism in the same way, we simply fix $\epsilon = 1$ for all experiments and $\delta = 0.0001$ for experiments with (ϵ, δ)-differential privacy. All relationships between techniques hold for other settings of the privacy parameters and any input data.

Overall, these results indicate that the matrix mechanism can greatly improve the baseline methods even without optimal strategy selection. Further, the results of Nikolov et al. [24] show that the minimum error of the matrix mechanism is within a factor of $O(\log^2 n)$ to the minimum noise of any data-independent algorithm under (ϵ, δ)-differential privacy.

8 The matrix mechanism with nonnegativity constraints

As stated in Sect. 4, we have analyzed the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ over differentially private algorithms $\mathcal{K}$ for which the amount of noise added by $\mathcal{K}$ does not change with different underlying datasets. In addition, according to Proposition 7, the noise added to the answer to the workload $\mathbf{W}$ only depends on the workload $\mathbf{W}$ and the strategy $\mathbf{A}$. Such properties are fundamental to our closed-form error formulae (4) and (5). On the other hand, such properties prevent the matrix mechanism from adding any further constraints to the output data vector $\hat{\mathbf{x}}$ or query answers $\mathbf{W}\hat{\mathbf{x}}$. Consequently, using the matrix mechanism directly cannot take advantage of any properties of the underlying database. In this section, we extend the matrix mechanism to take into account non-

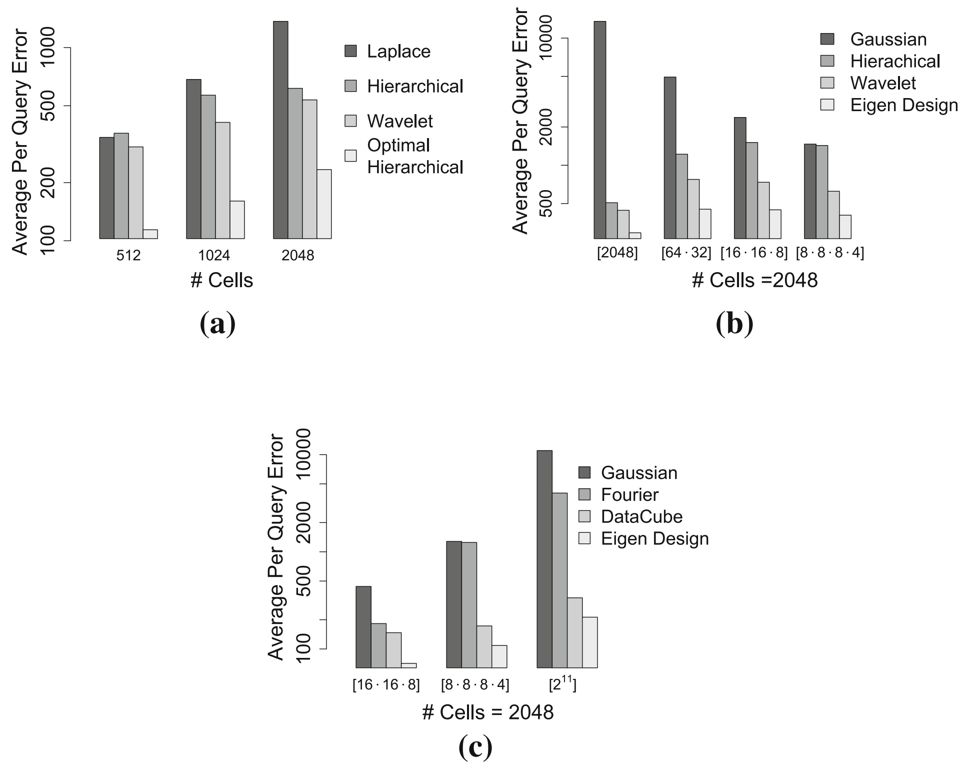

Fig. 8 Comparison of error between the matrix mechanism with different strategies and baseline mechanisms, where $\epsilon = 1$ and $\delta = 0.0001$ (for (ϵ, δ)-differential privacy only). **a** Average error on all 1-dimensional range queries under ϵ-differential privacy. **b** Average error on all range queries over cells of different dimensions under (ϵ, δ)-differential privacy. **c** Average error on all 1- and 2-way marginal queries over cells of different dimensions under (ϵ, δ)-differential privacy

negativity constraints that must hold on the original database vector $\mathbf{x}$.

8.1 The nonnegativity constraint

Given a data vector $\mathbf{x}$, since the number in each entry of $\mathbf{x}$ is the number of tuples that satisfy a certain cell condition, each entry of $\mathbf{x}$ must be nonnegative. Since the matrix mechanism only provides the estimated query answers instead of providing $\hat{\mathbf{x}}$ directly, the nonnegative constraint in the matrix mechanism can be formulated as the existence of a nonnegative data vector $\hat{\mathbf{x}}$ such that vector $\mathbf{W}\hat{\mathbf{x}}$ is the same as the vector of estimated query answers.

Definition 16 For a given workload $\mathbf{W}$, an answer $\hat{\mathbf{y}}_{\mathbf{W}}$ to $\mathbf{W}$ satisfies the nonnegativity constraint if and only if there exists a nonnegative data vector $\hat{\mathbf{x}}$ such that $\hat{\mathbf{y}}_{\mathbf{W}} = \mathbf{W}\hat{\mathbf{x}}$.

Using the matrix mechanism as described in Sect. 4 cannot guarantee nonnegative answers to the workload queries. The most straightforward solution is to compute an estimated data vector $\hat{\mathbf{x}} = \mathbf{A}\hat{\mathbf{y}}$, round all negative entries of $\hat{\mathbf{x}}$ up to zero, and compute the answer to $\mathbf{W}$ using the new data vector. We present two additional approaches in the next section that enforce the nonnegativity constraint. Both approaches rely on quadratic programming with the nonnegativity constraint, and the major difference between them is the goal of optimization. The first approach uses the standard nonnegative least square estimator which aims to minimize the distance to the data vector, while the second approach minimizes the distance to the workload answers.

8.2 Enforcing the nonnegativity constraint

To enforce the nonnegativity constraint, let us first revisit how the matrix mechanism works. According to Proposition 7,

$$\begin{aligned}\mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{W}, \mathbf{x}) &= \mathbf{W}\mathbf{A}^+\left(\mathbf{A}\mathbf{x} + ||\mathbf{A}||\tilde{\mathbf{b}}\right)\\ &= \mathbf{W}\mathbf{x} + \mathbf{W}\mathbf{A}^+||\mathbf{A}||\tilde{\mathbf{b}}.\end{aligned}$$

If the nonnegative data vector $\hat{\mathbf{x}}$ such that $\mathbf{W}\hat{\mathbf{x}} = \mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{W}, \mathbf{x})$ does not exist, one should derive a nonnegative data vector $\hat{\mathbf{x}}$ such that $\mathbf{W}\hat{\mathbf{x}}$ is close to $\mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{W}, \mathbf{x})$.

8.2.1 The nonnegative least squares estimator

According to Proposition 17, for any $\mathbf{y} = \mathbf{A}\mathbf{x} + \tilde{\mathbf{b}}$, where entries of $\tilde{\mathbf{b}}$ are i.i.d random variables, $\mathbf{A}^+\mathbf{y}$ satisfies

$$\forall \mathbf{x}' \quad ||\mathbf{y} - \mathbf{A}\mathbf{A}^+\mathbf{y}||_2 \leq ||\mathbf{y} - \mathbf{A}\mathbf{x}'||_2.$$

Hence, $\mathbf{A}^+\mathbf{y}$ is a least square estimator of $\mathbf{x}$. Now, we add an additional nonnegativity constraint to this process and use the nonnegative least square estimator to take the place of $\mathbf{A}^+\mathbf{y}$, which can be computed using the following optimization formulation:

Given: $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{y} \in \mathbb{R}^m$.
Minimize: $||\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}||_2^2$.
Subject to: $\hat{\mathbf{x}} \geq 0$.

With the vector $\hat{\mathbf{x}}$ given by the nonnegative least square estimator, one can compute answers to the workload $\mathbf{W}$ directly as $\mathbf{W}\hat{\mathbf{x}}$, which automatically satisfies the nonnegativity constraint. Solving for the nonnegative solution that minimizes squared error is a well-known optimization problem with a variety of solutions and many available implementations.

Example 9 Let $\mathbf{A} = \mathbf{Y}_4$ and $\mathbf{x} = (0, 2, 1, 3)^T$, and $\mathbf{y} = (5.85, -3.51, 0.59, -6.53)^T$.

– Using the least square estimator $\mathbf{A}^+\mathbf{y}$, we can compute an estimated data vector as (0.88, 0.29,-0.93, 5.60).
– Using the nonnegative least square estimator above, the estimated data vector is (0.88, 0.29, 0, 5.30).

8.2.2 Nonnegative least squares on the workload

One of the potential problems with using the nonnegative least square estimator above is the case in which the strategy contains non-overlapping queries whose true answers are small compared with the magnitude of noise. In such cases, the nonnegative least square estimator would eliminate all negative noise while keeping positive noise, which leads to an unnecessary amount of noise accumulation when computing the answers to the workload using the answers to the strategy. The extreme case is when $\mathbf{A} = \mathbf{I}_n$. Using nonnegative least square is equivalent to rounding up all negative entries of $\hat{\mathbf{x}}$ to 0. Since those negative entries correspond to negative noise, rounding them up to 0 leads to biased estimates and reduces the chance of canceling error by adding entries with positive and negative noise. Hence, using nonnegative least squares in such cases often leads to a very poorly estimated answer to the workload.

In order to address this weakness, we can keep the original framework of the matrix mechanism to answer the workload. We then post-process the answers to the workload by using the nonnegative least square, described as follows:

Given: $\mathbf{W} \in \mathbb{R}^{m_1 \times n}$, $\mathbf{A} \in \mathbb{R}^{m_2 \times n}$, $\mathbf{y} \in \mathbb{R}_2^m$.
Minimize: $||\mathbf{W}\mathbf{A}^+\mathbf{y} - \mathbf{W}\hat{\mathbf{x}}||_2^2$.
Subject to: $\hat{\mathbf{x}} \geq 0$.

Example 10 As in Example 9, let $\mathbf{A} = \mathbf{Y}_4, \mathbf{x} = (0, 2, 1, 3)^T$, and $\mathbf{y} = (5.85, -3.51, 0.59, -6.53)^T$. Further, let $\mathbf{W} = \mathbf{W}_R$. Using the nonnegative least square with the workload, the estimated data vector yields $(0.76, 0, 0, 5.08)^T$.

8.3 Comparing different approaches

Since neither of the approaches mentioned above relies on the least square estimator, their error cannot be computed via error formula (4) or (5). Therefore, in this section, we empirically compare the effect of these two approaches, along with the naive rounding approach.

Two datasets are considered in our experiments, which are the same datasets used in [19]. The Nettrace dataset contains the IP-level network traces from a major university. The Search_Logs dataset includes the search query logs that report the frequency of the phrase "Obama" from 2004 to 2010. Both datasets are aggregated so that each has 512 cells. The Nettrace dataset is more sparse and has smaller average cell count on nonzero cells. We include three strategies in the experiments: the hierarchical strategy ($\mathbf{H}_n$) [19], the privelet strategy ($\mathbf{Y}_n$) [28], and the identity strategy ($\mathbf{I}_n$).

The results are shown in Fig. 9, where both the nonnegative least square estimator (NNLS) and the nonnegative least square on the workload (NNLS on $\mathbf{W}$) are compared with the original matrix mechanism with the least square estimator (LS). The workload of all range queries is fixed, and all three approaches are applied to both datasets and the three different strategies mentioned above. The $\mathcal{K}$ is set to be the Laplace mechanism, and ϵ is varied from 0.001 to 1.

As the figure demonstrates, NNLS on $\mathbf{W}$ always introduces lower error than both NNLS and LS. In particular, the error of NNLS on $\mathbf{W}$ is much smaller on the Nettrace dataset: as much as 10 times smaller than both NNLS and LS. Meanwhile, the performance of NNLS heavily depends on the strategy: It has lower error than LS when using the hierarchical strategy, similar error as LS when using the privelet strategy, and much worse error than LS when using the identity strategy. Intuitively, such differences may come from

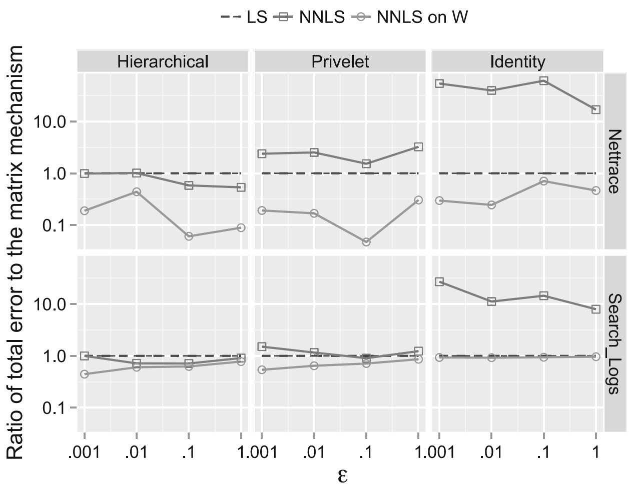

Fig. 9 Error comparison among the matrix mechanism (LS), the nonnegative least square estimator (NNLS), and the nonnegative least square on the workload (NNLS on W)

the different number of strategy queries that must be combined to answer a workload query: The number of strategy queries to be combined using the hierarchical strategy, which is $O(\log n)$, is much less than the number when using the identity strategy, which is $O(n)$. The results for the naive approach of rounding negative entries in $\mathbf{x}$ to zero are omitted in this figure since it is worse than NNLS by more than two orders of magnitude (except when the strategy is $\mathbf{I}_n$, in which case the round up approach is exactly the same as NNLS).

The natural formulation of the problem of finding an optimal strategy using nonnegativity constraints will contain both the workload and the data vector, since the error depends on both. However, solving for the best strategy using this formulation violates differential privacy since one needs to access the data vector during the process. Because it is difficult to find a differentially private formulation of the problem, our current approach is to find a strategy without the nonnegativity constraint (i.e., using the techniques of Sec. 7) and use the nonnegative constraint to reduce the error afterward. The theoretical analysis of the matrix mechanism with nonnegativity constraints and the formulation of the corresponding optimization problem are left as future work.

9 Related work

Since its original definition, differential privacy [12] has been the subject of considerable research, as outlined in recent surveys [7–9].

Two prior techniques specifically tailored to range queries can be viewed as special instances of the matrix mechanism. The first uses a wavelet transformation [28]; the second uses a hierarchical set of queries followed by inference [19]. Meanwhile, other works that designed algorithms that answer marginal queries can also be considered as instances of the matrix mechanism. Each of those works designs special strategies for a specific class of workloads. Barak et al. [2] studied answering low-order marginal queries using subsets of the Fourier basis; Ding et al. [6] considered a special collection of marginal queries, called data cubes, which are answered by a subset of all data cube queries selected via a greedy algorithm. The algorithm adapts a known approximation algorithm for the subset sum problem and cannot be applied to general linear queries.

As follow-up works to the matrix mechanism, an adaptive algorithm that generates strategies for any workload under (ϵ, δ)-differential privacy was presented in [21]. The algorithm generates different strategies by weighting queries that consist of the singular vectors of the query workload. An empirical study demonstrates that the strategies produced by this algorithm outperform previously designed algorithms on various workloads [2,6,19,28]. The case of supporting low-rank workloads under ϵ-differential privacy are discussed by Yuan et al. [32] when the number of queries is much smaller than the size of the domain. Though represented in a different form, the inference matrix used in [32] is exactly the same as the matrix mechanism. In order to avoid the hardness of solving the optimization problem under the matrix mechanism, Yaroslavtsev et al. [31] demonstrated an approach that has a similar form of the matrix mechanism, but relies on a fixed "recovery" matrix instead of the least square inference as the last step of the matrix mechanism. Such a simplification leads to a much easier optimization formulation, though inference with the recovery matrix introduces more noise than inference with the least squares.

The matrix mechanism and its variations mentioned above do not change across different datasets: As long as the workload stays the same, the same set of queries will be answered and the answer to the workload is derived using the same process, either by the least square estimator or by the recovery matrix. The recent work of Nikolov et al. [24] further demonstrates that the minimum error of the matrix mechanism is within a factor of $O(\log^2 n)$ to the optimal error for any data-independent mechanisms under (ϵ, δ)-differential privacy.

However, data independence does not hold for the discussion in Sect. 8, since the nonnegative least square estimator depends on the input data. Similarly, many recently proposed techniques for linear query answering also depend on the underlying database because they exploit properties of the input database to determine queries to be answered and/or the noise to be added to query answers. Practical algorithms are presented to answer linear queries on 1-dimensional databases [1,30] and 2-dimensional databases [4,29], respectively. Hardt et al. [16] designed an algorithm that supports any linear queries under differential privacy. Further, there are more data-dependent approaches from the theory community. Those approaches are both data-aware and interactive and lead to asymptotically smaller error than the matrix mechanism over sparse databases by analyzing the properties of the underlying database. The median mechanism [27] maintains a set of database instances that consist of historical query answers. The new query is either answered by the maintained set of databases or by the original database, determined in a differentially private way. Dwork et al. [13] sampled linear queries in each step and modified the sample distribution with the new query answers. In [15,17], the authors recursively update the estimated data vector to reduce the error on linear queries. In each of those algorithms, asymptotic error bounds are provided. More generally, Dwork et al. provide an error bound using an arbitrary differentially private mechanism [11] but not specifically for linear counting queries. Theoretically, those approaches have better dependency on privacy parameters but may not be applicable in practice.

10 Conclusion

We have described the matrix mechanism, which derives answers to a workload of counting queries from the noisy answers to a different set of strategy queries. By designing the strategy queries for the workload, correlated sets of counting queries can be answered more accurately. We showed that the optimal strategy can be computed by iteratively solving a pair of semidefinite programs, and we investigated alternative derivation methods which account for nonnegativity constraints.

Future directions include extending the matrix mechanism for operation in an interactive (rather than batch) setting and understanding precisely the conditions under which optimal strategies in our framework match known lower bounds for the accuracy of differentially private algorithms. We are also interested in combining the matrix mechanism, which is effective for exploiting properties of the workload in a data-independent way, with algorithmic techniques that adapt to the input database.

Although the techniques in this paper have been shown to offer a significant reduction in absolute error when compared to baseline and competing methods, it remains to demonstrate the overall effectiveness of these privacy algorithms to downstream data analytics tasks. Such tasks may use a workload of linear queries as sufficient statistics for some other computation. To assess utility in a manner that is most relevant to the end-user, we would like to measure it in terms of the final use of the data.

Acknowledgments We appreciate the comments of each of the anonymous reviewers. Hay, Li, and Miklau were supported by the National Science Foundation under NSF Award Numbers IIS-0964094, CNS 1012748 and CNS-1409143. McGregor was supported by CCF-0953754. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

Appendix 1: Linear algebra fundamentals

In this section, we summarize the concepts and results in linear algebra and matrix analysis that are used throughout the paper. We use $\mathrm{diag}(d_1, \dots d_n)$ to indicate the $n \times n$ diagonal matrix with scalars d_i on the diagonal and $\mathbf{0}^{m\times n}$ to indicate a matrix of zeroes with m rows and n columns. Recall that for a matrix $\mathbf{A}$, $\mathbf{A}^T$ is its transpose and $\mathbf{A}^{-1}$ is its inverse. We say $\mathbf{A}$ is symmetric if $\mathbf{A}^T = \mathbf{A}$ and orthogonal if $\mathbf{A}^T = \mathbf{A}^{-1}$. The rank of a matrix $\mathbf{A}$, $\mathrm{rank}(\mathbf{A})$, is defined as the size of the largest set of linearly independent rows (or equivalently columns) of $\mathbf{A}$. We say a matrix is full row (column) rank if its rank is equal to the number of its rows (columns). In particular, $\mathbf{A}^{-1}$ exists if and only if $\mathbf{A}$ is a square matrix with full rank.

If matrix $\mathbf{A}$ is a square matrix, the trace of $\mathbf{A}$, denoted as $\mathrm{trace}(\mathbf{A})$, is the sum of entries on the main diagonal if $\mathbf{A}$. The trace of a matrix has a very important property: It is invariant under cyclic permutations; i.e, if matrix $\mathbf{A}_1$ has m columns and matrix $\mathbf{A}_3$ has m rows,

$$\mathrm{trace}(\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3) = \mathrm{trace}(\mathbf{A}_3\mathbf{A}_1\mathbf{A}_2)$$

Another concept that is related to the trace is the Frobenius norm. The Frobenius norm of $\mathbf{A}$ is denoted as $||\mathbf{A}||_F$ and defined as $\sqrt{\mathrm{trace}(\mathbf{A}^T\mathbf{A})}$, or, equivalently, the square root of the squared sum of all entries in $\mathbf{A}$.

Matrix decomposition is extensively used in the paper. We focus on two decompositions: eigenvalue decomposition and singular value decomposition. Given a matrix $\mathbf{A}$, the eigenvalue decomposition of $\mathbf{A}$ always exists when $\mathbf{A}$ is symmetric. It can be written as $\mathbf{A} = \mathbf{QDQ}^T$ where $\mathbf{Q}$ is an orthogonal matrix whose columns are eigenvectors of $\mathbf{A}$ and $\mathbf{D}$ is a diagonal matrix whose diagonal entries are eigenvalues of $\mathbf{A}$. The singular value decomposition of $\mathbf{A}$ always exists and has the form $\mathbf{A} = \mathbf{QDP}^T$ where $\mathbf{Q}$ and $\mathbf{P}$ are orthogonal matrices and $\mathbf{D}$ is a diagonal matrix padded with columns or rows of 0s.

We will also rely on the notion of the positive semidefinite matrix. A symmetric square matrix $\mathbf{A}$ is called positive semidefinite, denoted as $\mathbf{A} \succeq 0$, if for any vector $\mathbf{x}$, $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$. In particular, for any matrix $\mathbf{A}$, $\mathbf{A}^T\mathbf{A} \succeq 0$. Here, we present two equivalent conditions to positive semidefinite.

Proposition 15 *Given an $n \times n$ symmetric matrix $\mathbf{A}$, both of the following conditions are equivalent with $\mathbf{A} \succeq 0$.*

(i) *All the eigenvalues of $\mathbf{A}$ are nonnegative.*
(ii) *For any $1 \leq i_1 < \cdots < i_k \leq n$, the determinant of the matrix that consists of the intersection of the $i_1^{th}, \dots, i_k^{th}$ rows and $i_1^{th}, \dots, i_k^{th}$ columns of matrix $\mathbf{A}$ is nonnegative.*

In addition, we consider a generalization of the matrix inverse, called the Moore–Penrose pseudoinverse, which is defined as follows:

Definition 17 (Moore–Penrose Pseudoinverse [3]) Given a $m \times n$ matrix $\mathbf{A}$, a matrix $\mathbf{A}^+$ is the Moore–Penrose pseudoinverse of $\mathbf{A}$ if it satisfies each of the following:

$$\mathbf{AA}^+\mathbf{A} = \mathbf{A},\ \mathbf{A}^+\mathbf{AA}^+ = \mathbf{A}^+,\ (\mathbf{AA}^+)^T = \mathbf{AA}^+,\ (\mathbf{A}^+\mathbf{A})^T = \mathbf{A}^+\mathbf{A}.$$

The Moore–Penrose pseudoinverse is unique and can be computed using the singular value decomposition of a matrix.

Proposition 16 ([3]) *Given an $n \times n$ diagonal matrix $\mathbf{D}_0$, $\mathbf{D}_0^+ = \{d'_{ij}\}$ is an $n \times n$ diagonal matrix such that*

$$d'_{ij} = \begin{cases} 0 & d_{ij} = 0 \\ \frac{1}{d_{ij}} & d_{ij} \neq 0 \end{cases}$$

For an $m \times n$ matrix $\mathbf{D}$ consisting of a diagonal matrix $\mathbf{D}_0$ padding with columns (rows) of 0*s,* $\mathbf{D}^+$ *is an $n \times m$ matrix consisting of the diagonal matrix* $\mathbf{D}_0^+$ *with rows (columns) of* 0*s. Given a matrix* $\mathbf{A}$ *with singular value decomposition* $\mathbf{A} = \mathbf{QDP}^T$, $\mathbf{A}^+ = \mathbf{PD}^+\mathbf{Q}^T$.

When $\mathbf{A}$ has full column rank, $\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$. We include some important properties of the Moore–Penrose pseudoinverse in the following proposition.

Proposition 17 ([3]) *The Moore–Penrose pseudoinverse satisfies the following properties:*

1. *Given any matrix* $\mathbf{A}$*, there exists a unique matrix that is the Moore–Penrose pseudoinverse of* $\mathbf{A}$.
2. *Given a vector* $\mathbf{y}$*, we have* $||\mathbf{y} - \mathbf{Ax}||_2 \geq ||\mathbf{y} - \mathbf{AA}^+\mathbf{y}||_2$ *for any vector* $\mathbf{x}$.
3. *For any satisfiable linear system* $\mathbf{BA} = \mathbf{W}$, $\mathbf{WA}^+$ *is a solution to the linear system and* $||\mathbf{WA}^+||_F \leq ||\mathbf{B}||_F$ *for any solution* $\mathbf{B}$ *to the linear system.*

Appendix 2: Proofs

Proofs from Section 4

Proposition 6 *The matrix mechanism* $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ *inherits the privacy guarantee of* $\mathcal{K}$ *and is unbiased if* $\mathcal{K}$ *is unbiased.*

Proof According to Eq. (1), the matrix mechanism can be considered to post-process the output of $\mathcal{K}(\mathbf{A}, \mathbf{x})$, without using $\mathbf{x}$, and hence shares the same privacy guarantee with $\mathcal{K}(\mathbf{A}, \mathbf{x})$. In addition, since $\mathbf{WA}^+\mathbf{A} = \mathbf{W}$, we have:

$$\begin{aligned}\mathbb{E}[\mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{A}, \mathbf{x})] &= \mathbb{E}[\mathbf{WA}^+\mathcal{K}(\mathbf{A}, \mathbf{x})] \\ &= \mathbf{WA}^+\mathbb{E}[\mathcal{K}(\mathbf{A}, \mathbf{x})] = \mathbf{WA}^+\mathbf{Ax} = \mathbf{Wx}.\end{aligned}$$

□

Proofs from Section 5

Corollary 1 *Given two query strategies* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *that are profile equivalent, for any query* $\mathbf{W}$, $\mathbf{A}_1$ *supports* $\mathbf{W}$ *if and only if* $\mathbf{A}_2$ *supports* $\mathbf{W}$*. Furthermore, there exists a nonzero constant c such that given a differentially private algorithm* $\mathcal{K}$*, for any workload query* $\mathbf{W}$ *that* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *support,* $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}) = c \cdot \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W})$.

Proof Given a query workload $\mathbf{W}$, if $\mathbf{A}_1$ supports $\mathbf{W}$, there exists a matrix $\mathbf{X}$ such that $\mathbf{W} = \mathbf{XA}_1$. According to Proposition 11(iii), $\mathbf{A}_1$ and $\mathbf{A}_2$ are profile equivalent if and only if there exists a nonzero constant c and an orthogonal matrix $\mathbf{Q}$ such that $\mathbf{A}_1 = c \cdot \mathbf{QA}_2$. Then, $c \cdot \mathbf{XQ}$ satisfies $\mathbf{W} = c \cdot \mathbf{XQA}_2$, and therefore, $\mathbf{A}_2$ supports $\mathbf{W}$ as well.

The definition of profile equivalence indicates that there is a constant c' such that $(\mathbf{A}_1^T\mathbf{A}_1)^+ = c' \cdot (\mathbf{A}_2^T\mathbf{A}_2)^+$. Thus, for any query workload that $\mathbf{A}_1$ supports:

$$\begin{aligned}\frac{\textsc{TotalError}_{\mathcal{K},A_1}(W)}{\textsc{TotalError}_{\mathcal{K},A_2}(W)} &= \frac{||\mathbf{A}_1||^2||\mathbf{WA}_1^+||_F^2}{||\mathbf{A}_2||^2||\mathbf{WA}_2^+||_F^2} \\ &= \frac{||\mathbf{A}_1||^2\text{trace}(\mathbf{W}(\mathbf{A}_1^T\mathbf{A}_1)^+\mathbf{W}^T)}{||\mathbf{A}_2||^2\text{trace}(\mathbf{W}(\mathbf{A}_2^T\mathbf{A}_2)^+\mathbf{W}^T)} \\ &= c'\frac{||\mathbf{A}_1||^2}{||\mathbf{A}_2||^2},\end{aligned}$$

where the ratio is a value that is independent of $\mathbf{W}$. □

10.1 Proofs from Section 6

Corollary 2 *For any linear counting query* $\mathbf{w}$ *and differentially private mechanism* $\mathcal{K}$,

$$\frac{1}{2}\textsc{Error}_{\mathcal{K},\mathbf{Y}}(\mathbf{w}) \leq \textsc{Error}_{\mathcal{K},\mathbf{H}}(\mathbf{w}) \leq 2\textsc{Error}_{\mathcal{K},\mathbf{Y}}(\mathbf{w}).$$

Proof According to Theorem 1, let $\mathbf{H}'_n$ be the matrix that results from removing the row of all 1s from matrix $\begin{bmatrix}\mathbf{H}_n \\ \mathbf{I}_n\end{bmatrix}$. Since $\mathbf{H}'_n$ and $\mathbf{Y}_n$ are equivalent strategies under both ϵ- and (ϵ, δ)-differentially private mechanisms, it is sufficient to prove that for any linear counting query $\mathbf{w}$,

$$\frac{1}{2}\textsc{Error}_{\mathcal{K},\mathbf{H}'_n}(\mathbf{w}) \leq \textsc{Error}_{\mathcal{K},\mathbf{H}_n}(\mathbf{w}) \leq 2\textsc{Error}_{\mathcal{K},\mathbf{H}'_n}(\mathbf{w}).$$

Let $\mathbf{v} = \mathbf{wH}_n^+$ and $\mathbf{v}'$ be a vector such that

$$v'_i = \begin{cases} v_1 & 1 \leq i \leq 2 \\ v_{i-1} & 3 \leq i \leq 2n \\ 0 & 2n+1 \leq i \leq 3n \end{cases}.$$

One can verify that $\mathbf{v}'\mathbf{H}'_n = \mathbf{vH}_n = \mathbf{w}$. Since

$$||\mathbf{wH}'^+_n||_F \leq ||\mathbf{v}'||_F \leq 2||\mathbf{v}||_F = ||\mathbf{wH}_n^+||_F,$$

noticing that $||\mathbf{H}_n||_1 = ||\mathbf{H}'_n||_1$ and $||\mathbf{H}_n||_2 = ||\mathbf{H}'_n||_2$, $\textsc{Error}_{\mathcal{K},\mathbf{H}'_n}(\mathbf{w}) \leq 2\textsc{Error}_{\mathcal{K},\mathbf{H}_n}(\mathbf{w})$.

On the other hand, $\mathbf{H}'_n$ contains two copies of queries $\mathbf{I}_n$, which is equivalent to reduce the error on those queries by a factor of 2. Noticing all other queries in $\mathbf{H}'_n$ are contained in $\mathbf{H}_n$, we have $\textsc{Error}_{\mathcal{K},\mathbf{H}_n}(\mathbf{w}) \leq 2\textsc{Error}_{\mathcal{K},\mathbf{H}'_n}(\mathbf{w})$. □

Corollary 3 (Cell condition simplification) *Given a workload* $\mathbf{W}_1 \subseteq \mathbf{W}_R$ *with m queries and its corresponding* *list of cell conditions, Φ_1, there exists a workload $\mathbf{W}_2$ and a list of cell conditions, Φ_2, such that $(\mathbf{W}_1, \Phi_1) \equiv (\mathbf{W}_2, \Phi_2)$, $|\Phi_2| \leq 2m - 1$.*

Proof Since $\mathbf{W}_1 \subseteq \mathbf{W}_R$, let $b_1, \ldots, b_{2m}$ be the boundaries of queries in $\mathbf{W}_1$ such that $b_1 \leq \cdots \leq b_{2m}$. For any query $q \in W_1$, it contains either all cells between b_i and b_{i+1} or none of them. Therefore, columns $b_i, \ldots, b_{i+1} - 1$ are exactly the same. According to Theorem 7, removing columns $b_i + 1, \ldots, b_{i+1} - 1$ and merging the cell conditions of $b_i, \ldots, b_{i+1} - 1$ results in a workload and a list of cell conditions that is equivalent with $(\mathbf{W}_1, \Phi_1)$. Thus, merging cells between $b_i, \ldots, b_{i+1} - 1$ for $i = 1, \ldots, 2m - 1$ and discarding the rest of the cells gives a workload $\mathbf{W}_2$ and its corresponding list of cell conditions Φ_2, such that $(\mathbf{W}_2, \Phi_2) \equiv (\mathbf{W}_1, \Phi_1)$ and $|\Phi_2| \leq 2m - 1$. □

References

1. Ács, G., Castelluccia, C., Chen, R.: Differentially private histogram publishing through lossy compression. In: ICDM, pp. 1–10 (2012)
2. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In: PODS (2007)
3. Ben-Israel, A., Greville, T.: Generalized Inverses: Theory and Applications, vol. 15. Springer, Berlin (2003)
4. Cormode, G., Procopiuc, M., Shen, E., Srivastava, D., Yu, T.: Differentially private spatial decompositions. In: ICDE (2012)
5. Dattorro, J.: Convex Optimization & Euclidean Distance Geometry. Meboo Publishing, USA (2005)
6. Ding, B., Winslett, M., Han, J., Li, Z.: Differentially private data cubes: optimizing noise sources and consistency. In: SIGMOD, pp. 217–228 (2011)
7. Dwork, C.: Differential privacy: a survey of results. In: TAMC (2008)
8. Dwork, C.: The differential privacy frontier. In: TCC (2009)
9. Dwork, C.: A firm foundation for private data analysis. Commun. ACM **54**(1), 86–95 (2011)
10. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: privacy via distributed noise generation. In: EUROCRYPT, pp. 486–503 (2006)
11. Dwork, C., Naor, M., Reingold, O., Rothblum, G., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: STOC, pp. 381–390 (2009)
12. Dwork, C., Nissim, F.M.K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: TCC (2006)
13. Dwork, C., Rothblum, G.N., Vadhan, S.P.: Boosting and differential privacy. In: FOCS, pp. 51–60 (2010)
14. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. In: STOC (2009)
15. Gupta, A., Roth, A., Ullman, J.: Iterative constructions and private data release. In: TCC, pp. 339–356 (2012)
16. Hardt, M., Ligett, K., McSherry, F.: A simple and practical algorithm for differentially private data release. In: NIPS, pp. 2348–2356 (2012)
17. Hardt, M., Rothblum, G.: A multiplicative weights mechanism for privacy-preserving data analysis. In: FOCS, pp. 61–70 (2010)
18. Hardt, M., Talwar, K.: On the geometry of differential privacy. In: STOC, pp. 705–714 (2010)
19. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially-private histograms through consistency. PVLDB **3**(1–2), 1021–1032 (2010)
20. Li, C., Hay, M., Rastogi, V., Miklau, G., McGregor, A.: Optimizing linear counting queries under differential privacy. In: PODS, pp. 123–134 (2010)
21. Li, C., Miklau, G.: An adaptive mechanism for accurate query answering under differential privacy. PVLDB **5**(6), 514–525 (2012)
22. McSherry, F., Mironov, I.: Differentially private recommender systems: building privacy into the netflix prize contenders. In: SIGKDD (2009)
23. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: SIGMOD, pp. 19–30 (2009)
24. Nikolov, A., Talwar, K., Zhang, L.: The geometry of differential privacy: the sparse and approximate cases. In: STOC (2013)
25. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: STOC, pp. 75–84 (2007)
26. Qardaji, W.H., Yang, W., Li, N.: Understanding hierarchical methods for differentially private histograms. PVLDB **6**(14), 1954–1965 (2013)
27. Roth, A., Roughgarden, T.: Interactive privacy via the median mechanism. In: STOC, pp. 765–774 (2010)
28. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. In: ICDE, pp. 225–236 (2010)
29. Xiao, Y., Gardner, J.J., Xiong, L.: Dpcube: Releasing differentially private data cubes for health information. In: ICDE, pp. 1305–1308 (2012)
30. Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G., Winslett, M.: Differentially private histogram publication. VLDB J **22**(6), 797–822 (2013)
31. Yaroslavtsev, G., Cormode, G., Procopiuc, C.M., Srivastava, D.: Accurate and efficient private release of datacubes and contingency tables. In: ICDE (2013)
32. Yuan, G., Zhang, Z., Winslett, M., Xiao, X., Yang, Y., Hao, Z.: Low-rank mechanism: optimizing batch queries under differential privacy. In: VLDB (2012)