SIAM J. Sci. Comput.
Vol. 36, No. 5, pp. S78–S110

QUANTILE REGRESSION FOR LARGE-SCALE APPLICATIONS*

JIYAN YANG†, XIANGRUI MENG‡, AND MICHAEL W. MAHONEY§

Abstract. Quantile regression is a method to estimate the quantiles of the conditional distribution of a response variable, and as such it permits a much more accurate portrayal of the relationship between the response variable and observed covariates than methods such as least-squares or least absolute deviations regression. It can be expressed as a linear program, and, with appropriate preprocessing, interior-point methods can be used to find a solution for moderately large problems. Dealing with very large problems, e.g., involving data up to and beyond the terabyte regime, remains a challenge. Here, we present a randomized algorithm that runs in nearly linear time in the size of the input and that, with constant probability, computes a $(1+\epsilon)$ approximate solution to an arbitrary quantile regression problem. As a key step, our algorithm computes a low-distortion subspace-preserving embedding with respect to the loss function of quantile regression. Our empirical evaluation illustrates that our algorithm is competitive with the best previous work on small to medium-sized problems, and that in addition it can be implemented in MapReduce-like environments and applied to terabyte-sized problems.

Key words. quantile regression, random sampling algorithms, massive data set

AMS subject classification. 68W20

DOI. 10.1137/130919258

1. Introduction.

Quantile regression is a method to estimate the quantiles of the conditional distribution of a response variable, expressed as functions of observed covariates [8], in a manner analogous to the way in which least-squares regression estimates the conditional mean. The least absolute deviations regression (i.e., ℓ_1 regression) is a special case of quantile regression that involves computing the median of the conditional distribution. In contrast with ℓ_1 regression and the more popular ℓ_2 or least-squares regression, quantile regression involves minimizing asymmetrically weighted absolute residuals. Doing so, however, permits a much more accurate portrayal of the relationship between the response variable and observed covariates, and it is more appropriate in certain non-Gaussian settings. For these reasons, quantile regression has found applications in many areas, e.g., survival analysis and economics [2, 10, 3]. As with ℓ_1 regression, the quantile regression problem can be formulated as a linear programming problem, and thus simplex or interior-point methods can be applied [9, 15, 14]. Most of these methods are efficient only for problems of small to moderate size, and thus to solve very-large-scale quantile regression problems more reliably and efficiently, we need new computational techniques.

In this paper, we provide a fast algorithm to compute a $(1+\epsilon)$ relative-error approximate solution to the overconstrained quantile regression problem. Our algorithm constructs a low-distortion subspace embedding of the form that has been

*Received by the editors May 1, 2013; accepted for publication (in revised form) December 17, 2013; published electronically October 30, 2014. A conference version of this paper appears under the same title in *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, GA, 2013 [18].
http://www.siam.org/journals/sisc/36-5/91925.html

†ICME, Stanford University, Stanford, CA 94305 (jiyan@stanford.edu).
‡LinkedIn, Mountain View, CA 94043 (ximeng@linkedin.com).
§International Computer Science Institute and Department of Statistics, University of California at Berkeley, Berkeley, CA 94720 (mmahoney@icsi.berkeley.edu).

used in recent developments in randomized algorithms for matrices and large-scale data problems, and our algorithm runs in time that is nearly linear in the number of nonzeros in the input data.

In more detail, recall that a quantile regression problem can be specified by a (design) matrix $A \in \mathbb{R}^{n\times d}$, a (response) vector $b \in \mathbb{R}^n$, and a parameter $\tau \in (0, 1)$, in which case the quantile regression problem can be solved via the optimization problem

$$\text{minimize}_{x\in\mathbb{R}^d} \quad \rho_\tau(b - Ax), \tag{1.1}$$

where $\rho_\tau(y) = \sum_{i=1}^{d} \rho_\tau(y_i)$, for $y \in \mathbb{R}^d$, where

$$\rho_\tau(z) = \begin{cases} \tau z, & z \geq 0; \\ (\tau - 1)z, & z < 0, \end{cases} \tag{1.2}$$

for $z \in \mathbb{R}$, is the corresponding loss function. In the remainder of this paper, we will use A to denote the augmented matrix $\begin{bmatrix} b & -A \end{bmatrix}$, and we will consider $A \in \mathbb{R}^{n\times d}$. With this notation, the quantile regression problem of (1.1) can equivalently be expressed as a constrained optimization problem with a single linear constraint,

$$\text{minimize}_{x\in\mathcal{C}} \quad \rho_\tau(Ax), \tag{1.3}$$

where $\mathcal{C} = \{x \in \mathbb{R}^d \mid c^T x = 1\}$ and c is a unit vector with the first coordinate set to be 1. The reasons we want to switch from (1.1) to (1.3) are as follows. First, it is for notational simplicity in the presentation of our theorems and algorithms. Second, all the results about low-distortion or $(1 \pm \epsilon)$-subspace embedding in this paper hold for any $x \in \mathbb{R}^d$,

$$(1/\kappa_1)\|Ax\|_1 \leq \|\Pi Ax\|_1 \leq \kappa_2\|Ax\|_1.$$

In particular, we can consider x in some specific subspace of $\mathbb{R}^d$. For example, in our case, $x \in \mathcal{C}$. Then, the equation above is equivalent to the following:

$$(1/\kappa_1)\|b - Ax\|_1 \leq \|\Pi b - \Pi Ax\|_1 \leq \kappa_2\|b - Ax\|_1.$$

Therefore, using notation Ax with x in some constraint is a more general form of expression. We will focus on very overconstrained problems with size $n \gg d$.

Our main algorithm depends on a technical result, presented as Lemma 3.1, which is of independent interest. Let $A \in \mathbb{R}^{n\times d}$ be an input matrix, and let $S \in \mathbb{R}^{s\times n}$ be a random sampling matrix constructed based on the importance sampling probabilities

$$p_i = \min\{1, s \cdot \|U_{(i)}\|_1/\|U\|_1\},$$

where $\|\cdot\|_1$ is the elementwise ℓ_1 norm and where $U_{(i)}$ is the ith row of an ℓ_1 well-conditioned basis U for the range of A (see Definition 2.4 and Proposition 3.3). Then, Lemma 3.1 states that for a sampling complexity s that depends on d but is independent of n,

$$(1 - \epsilon)\rho_\tau(Ax) \leq \rho_\tau(SAx) \leq (1 + \epsilon)\rho_\tau(Ax)$$

will be satisfied for every $x \in \mathbb{R}^d$.

Although one could use, e.g., the algorithm of [6] to compute such a well-conditioned basis U and then "read off" the ℓ_1 norm of the rows of U, doing so would be much slower than the time allotted by our main algorithm. As Lemma 3.1 enables us to leverage the fast quantile regression theory and the algorithms developed for ℓ_1 regression, we provide two sets of additional results, most of which are built from the previous work. First, we describe three algorithms (Algorithm 1, Algorithm 2, and Algorithm 3) for computing an implicit representation of a well-conditioned basis; second, we describe an algorithm (Algorithm 4) for approximating the ℓ_1 norm of the rows of the well-conditioned basis from that implicit representation. For each of these algorithms, we prove quality-of-approximation bounds in quantile regression problems, and we show that they run in nearly "input-sparsity" time, i.e., in $\mathcal{O}(\text{nnz}(A) \cdot \log n)$ time, where $\text{nnz}(A)$ is the number of nonzero elements of A, plus lower-order terms. These lower-order terms depend on the time to solve the subproblem we construct, and they depend on the smaller dimension d but not on the larger dimension n. Although of less interest in theory, these lower-order terms can be important in practice, as our empirical evaluation will demonstrate.

We should note that of the three algorithms for computing a well-conditioned basis, the first two appear in [13] and are stated here for completeness; the third algorithm, which is new to this paper, is *not* uniformly better than either of the two previous algorithms with respect to either condition number or running time. (In particular, Algorithm 1 has slightly better running time, and Algorithm 2 has slightly better conditioning properties.) Our new conditioning algorithm is, however, only slightly worse than the better of the two previous algorithms with respect to each of those two measures. Because of the trade-offs involved in implementing quantile regression algorithms in practical settings, our empirical results show that by using a conditioning algorithm that is only slightly worse than the best previous conditioning algorithms for each of these two criteria, our new conditioning algorithm can lead to better results than either of the previous algorithms that was superior by only one of those criteria.

Given these results, our main algorithm for quantile regression is presented as Algorithm 5. Our main theorem for this algorithm, Theorem 3.4, states that with constant probability, this algorithm returns a $(1 + \epsilon)$-approximate solution to the quantile regression problem and that this solution can be obtained in $\mathcal{O}(\text{nnz}(A) \cdot \log n)$ time, plus the time for solving the subproblem (whose size is $\mathcal{O}(\mu d^3 \log(\mu/\epsilon)/\epsilon^2) \times d$, where $\mu = \frac{\tau}{1-\tau}$, independent of n, when $\tau \in [1/2, 1)$).

We also provide a detailed empirical evaluation of our main algorithm for quantile regression, including characterizing the quality of the solution as well as the running time, as a function of the high dimension n, the lower dimension d, the sampling complexity s, and the quantile parameter τ. Among other things, our empirical evaluation demonstrates that the output of our algorithm is highly accurate in terms of not only objective function value but also the actual solution quality (by the latter, we mean a norm of the difference between the exact solution to the full problem and the solution to the subproblem constructed by our algorithm), when compared with the exact quantile regression, as measured in three different norms. More specifically, our algorithm yields two-digit accuracy solution by sampling only, e.g., about 0.001% of a problem with size $2.5e9 \times 50$.[1] Our new conditioning algorithm outperforms other conditioning-based methods, and it permits much larger small dimension d than previous conditioning algorithms. In addition to evaluating our algorithm on

[1]We use this notation throughout; e.g., by $2.5e9 \times 50$, we mean that $n = 2.5 \times 10^9$ and $d = 50$.

moderately large data that can fit in RAM, we also show that our algorithm can be implemented in MapReduce-like environments and applied to computing the solution of terabyte-sized quantile regression problems.

The best previous algorithm for moderately large quantile regression problems is due to [15] and [14]. Their algorithm uses an interior-point method on a smaller problem that has been preprocessed by randomly sampling a subset of the data. Their preprocessing step involves predicting the sign of each $A_{(i)}x^* - b_i$, where $A_{(i)}$ and b_i are the ith row of the input matrix and the ith element of the response vector, respectively, and x^* is an optimal solution to the original problem. When compared with our approach, they compute an optimal solution, while we compute an approximate solution, but in worst-case analysis it can be shown that with high probability our algorithm is guaranteed to work, while their algorithm does not come with such guarantees. Also, the sampling complexity of their algorithm depends on the higher dimension n, while the number of samples required by our algorithm depends only on the lower dimension d, but our sampling is with respect to a carefully constructed nonuniform distribution, while they sample uniformly at random.

For a detailed overview of recent work on using randomized algorithms to compute approximate solutions for least-squares regression and related problems, see the recent review [12]. Most relevant for our work is the algorithm of [6] that constructs a well-conditioned basis by ellipsoid rounding and a subspace-preserving sampling matrix in order to approximate the solution of general ℓ_p regression problems, for $p \in [1, \infty)$, in roughly $\mathcal{O}(nd^5 \log n)$; the algorithms of [16] and [5] that use the "slow" and "fast" versions of the Cauchy transform to obtain a low-distortion ℓ_1 embedding matrix and solve the overconstrained ℓ_1 regression problem in $\mathcal{O}(nd^{1.376+})$ and $\mathcal{O}(nd \log n)$ time, respectively; and the algorithm of [13] that constructs low-distortion embeddings in input-sparsity time and uses those embeddings to construct a well-conditioned basis and approximate the solution of the overconstrained ℓ_1 regression problem in $\mathcal{O}(\text{nnz}(A) \cdot \log n + \text{poly}(d) \log(1/\epsilon)/\epsilon^2)$ time. In particular, we will use the two conditioning methods in [13], as well as our "improvement" of those two methods, for constructing ℓ_1 norm well-conditioned basis matrices in nearly input-sparsity time. In this work, we also demonstrate that such a well-conditioned basis in the ℓ_1 sense can be used to solve the overconstrained quantile regression problem.

2. Background and overview of conditioning methods.

2.1. Preliminaries.

We use $\|\cdot\|_1$ to denote the elementwise ℓ_1 norm for both vectors and matrices, and we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For any matrix A, $A_{(i)}$, and $A^{(j)}$ denote the ith row and the jth column of A, respectively; $\mathcal{A}$ denotes the column space of A. For simplicity, we assume A has full column rank, and we always assume that $\tau \geq \frac{1}{2}$. All the results hold for $\tau < \frac{1}{2}$ by simply switching the positions of τ and $1 - \tau$.

Although $\rho_\tau(\cdot)$, defined in (1.2), is not a norm, since the loss function does not have the positive linearity, it satisfies some "good" properties, as stated in the following lemma.

LEMMA 2.1. *Suppose that $\tau \geq \frac{1}{2}$. Then, for any $x, y \in \mathbb{R}^d, a \geq 0$, the following hold:*

1. $\rho_\tau(x + y) \leq \rho_\tau(x) + \rho_\tau(y)$;
2. $(1 - \tau)\|x\|_1 \leq \rho_\tau(x) \leq \tau\|x\|_1$;
3. $\rho_\tau(ax) = a\rho_\tau(x)$; *and*
4. $|\rho_\tau(x) - \rho_\tau(y)| \leq \tau\|x - y\|_1$.

Proof. It is trivial to prove every equality or inequality for x, y in one dimension. Then by the definition of $\rho_\tau(\cdot)$ for vectors, the inequalities and equalities hold for general x and y. ∎

To make our subsequent presentation self-contained, here we will provide a brief review of recent work on subspace embedding algorithms. We start with the definition of a low-distortion embedding matrix for $\mathcal{A}$ in terms of $\|\cdot\|_1$; see, e.g., [13].

DEFINITION 2.2 (low-distortion ℓ_1 subspace embedding). *Given $A \in \mathbb{R}^{n\times d}$, $\Pi \in \mathbb{R}^{r\times n}$ is a low-distortion embedding of $\mathcal{A}$ if $r = \text{poly}(d)$ and for all $x \in \mathbb{R}^d$,*

$$(1/\kappa_1)\|Ax\|_1 \le \|\Pi Ax\|_1 \le \kappa_2\|Ax\|_1.$$

where κ_1 and κ_2 are low-degree polynomials of d.

The following stronger notion of a $(1 \pm \epsilon)$-distortion subspace-preserving embedding will be crucial for our method. In this paper, the "measure functions" we will consider are $\|\cdot\|_1$ and $\rho_\tau(\cdot)$.

DEFINITION 2.3 ($(1 \pm \epsilon)$-distortion subspace-preserving embedding). *Given $A \in \mathbb{R}^{n\times d}$ and a measure function of vectors $f(\cdot)$, $S \in \mathbb{R}^{s\times n}$ is a $(1\pm\epsilon)$-distortion subspace-preserving matrix of $(\mathcal{A}, f(\cdot))$ if $s = \text{poly}(d)$ and for all $x \in \mathbb{R}^d$,*

$$(1-\epsilon)f(Ax) \le f(SAx) \le (1+\epsilon)f(Ax).$$

Furthermore, if S is a sampling matrix (one nonzero element per row in S), we call it a $(1 \pm \epsilon)$-distortion subspace-preserving sampling matrix.

In addition, the following notion, originally introduced by [4] and stated more precisely in [6], of a basis that is well-conditioned for the ℓ_1 norm will also be crucial for our method.

DEFINITION 2.4 ((α, β)-conditioning and well-conditioned basis). *Given $A \in \mathbb{R}^{n\times d}$, A is (α, β)-conditioned if $\|A\|_1 \le \alpha$ and for all $x \in \mathbb{R}^q, \|x\|_\infty \le \beta\|Ax\|_1$. Define $\kappa(A)$ as the minimum value of $\alpha\beta$ such that A is (α, β)-conditioned. We will say that a basis U of A is a well-conditioned basis if $\kappa = \kappa(U)$ is a polynomial in d, independent of n.*

For a low-distortion embedding matrix for $(\mathcal{A}, \|\cdot\|_1)$, we next state a fast construction algorithm that runs in input-sparsity time by applying the sparse Cauchy transform. This was originally proposed as Theorem 2 in [13].

LEMMA 2.5 (fast construction of low-distortion ℓ_1 subspace embedding matrix from [13]). *Given $A \in \mathbb{R}^{n\times d}$ with full column rank, let $\Pi_1 = SC \in \mathbb{R}^{r_1\times n}$, where $S \in \mathbb{R}^{r_1\times n}$ has each column chosen independently and uniformly from the r_1 standard basis vector of $\mathbb{R}^{r_1}$, and where $C \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonals chosen independently from Cauchy distribution. Set $r_1 = \omega d^5 \log^5 d$ with ω sufficiently large. Then, with a constant probability, we have*

$$1/\,\mathcal{O}(d^2 \log^2 d) \cdot \|Ax\|_1 \le \|\Pi_1 Ax\|_1 \le \mathcal{O}(d \log d) \cdot \|Ax\|_1 \;\; \forall x \in \mathbb{R}^d\,. \tag{2.1}$$

In addition, $\Pi_1 A$ can be computed in $\mathcal{O}(\text{nnz}(A))$ time.

Remark. This result has very recently been improved. In [17], the authors show that one can achieve a $\mathcal{O}(d^2 \log^2 d)$ distortion ℓ_1 subspace embedding matrix with embedding dimension $\mathcal{O}(d \log d)$ in $\text{nnz}(A)$ time by replacing Cauchy variables in the above lemma with exponential variables. Our theory can also be easily improved by using this improved lemma.

Next, we state a result for the fast construction of a $(1 \pm \epsilon)$-distortion subspace-preserving sampling matrix for $(\mathcal{A}, \|\cdot\|_1)$, from Theorem 5.4 in [5], with $p = 1$, as follows.

LEMMA 2.6 (fast construction of ℓ_1 sampling matrix from Theorem 5.4 in [5]). *Given a matrix $A \in \mathbb{R}^{n\times d}$ and a matrix $R \in \mathbb{R}^{d\times d}$ such that AR^{-1} is a well-conditioned basis for $\mathcal{A}$ with condition number κ, it takes $\mathcal{O}(\text{nnz}(A)\cdot \log n)$ time to compute a sampling matrix $S \in \mathbb{R}^{s\times n}$ with $s = \mathcal{O}(\kappa d \log(1/\epsilon)/\epsilon^2)$ such that with a constant probability, for any $x \in \mathbb{R}^d$,*

$$(1-\epsilon)\|Ax\|_1 \le \|SAx\|_1 \le (1+\epsilon)\|Ax\|_1.$$

We also cite the following lemma for finding a matrix R such that AR^{-1} is a well-conditioned basis, which is based on ellipsoidal rounding proposed in [5].

LEMMA 2.7 (fast ellipsoid rounding from [5]). *Given an $n \times d$ matrix A, by applying an ellipsoid rounding method, it takes at most $\mathcal{O}(nd^3 \log n)$ time to find a matrix $R \in \mathbb{R}^{d\times d}$ such that $\kappa(AR^{-1}) \le 2d^2$.*

Finally, two important ingredients for proving subspace preservation are γ-nets and tail inequalities. Suppose that Z is a point set and $\|\cdot\|$ is a metric on Z. A subset Z_γ is called a γ-net for some $\gamma > 0$ if for every $x \in Z$ there is a $y \in Z_\gamma$ such that $\|x - y\| \le \gamma$. It is well known that the unit ball of a d-dimensional subspace has a γ-net with size at most $(3/\gamma)^d$ [1]. Also, we will use the standard Bernstein inequality to prove concentration results for the sum of independent random variables.

LEMMA 2.8 (Bernstein inequality [1]). *Let $X_1, \ldots, X_n$ be independent random variables with zero-mean. Suppose that $|X_i| \le M$ for $i \in [n]$; then for any positive number t, we have*

$$\Pr\left[\sum_{i\in[n]} X_i > t\right] \le \exp\left(-\frac{t^2/2}{\sum_{i\in[n]} \mathbf{E}\, X_j^2 + Mt/3}\right).$$

2.2. Conditioning methods for ℓ_1 regression problems. Before presenting our main results, we start here by outlining the theory for conditioning for overconstrained ℓ_1 (and ℓ_p) regression problems.

The concept of a well-conditioned basis U (recall Definition 2.4) plays an important role in our algorithms, and thus in this subsection we will discuss several related conditioning methods. By a conditioning method, we mean an algorithm for finding, for an input matrix A, a well-conditioned basis, i.e., either finding a well-conditioned matrix U or finding a matrix R such that $U = AR^{-1}$ is well-conditioned. Many approaches have been proposed for conditioning. The two most important properties of these methods for our subsequent analysis are (1) the condition number $\kappa = \alpha\beta$ and (2) the running time to construct U (or R). The importance of the running time should be obvious, but the condition number directly determines the number of rows that we need to select, and thus it has an indirect effect on running time (via the time required to solve the subproblem). See Table 1 for a summary of the basic properties of the conditioning methods that will be discussed in this subsection.

In general, there are three basic ways for finding a matrix R such that $U = AR^{-1}$ is well-conditioned: those based on the QR factorization, those based on ellipsoid rounding, and those based on combining the two basic methods:

- Via QR factorization (QR). To obtain a well-conditioned basis, one can first construct a low-distortion ℓ_1 embedding matrix. By Definition 2.2, this means

Table 1

Summary of running time, condition number, and type of conditioning methods proposed recently. QR and ER refer, respectively, to methods based on the QR factorization and methods based on ellipsoid rounding, as discussed in the text. `QR_small` *and* `ER_small` *denote the running time for applying QR factorization and ellipsoid rounding, respectively, on a small matrix with size independent of* n*.*

Name	Running time	κ	Type
SC [16]	$\mathcal{O}(nd^2 \log d)$	$\mathcal{O}(d^{5/2} \log^{3/2} n)$	QR
FC [5]	$\mathcal{O}(nd \log d)$	$\mathcal{O}(d^{7/2} \log^{5/2} n)$	QR
Ellipsoid rounding [4]	$\mathcal{O}(nd^5 \log n)$	$d^{3/2}(d+1)^{1/2}$	ER
Fast ellipsoid rounding [5]	$\mathcal{O}(nd^3 \log n)$	$2d^2$	ER
SPC1 [13]	$\mathcal{O}(\text{nnz}(A))$	$\mathcal{O}(d^{\frac{13}{2}} \log^{\frac{11}{2}} d)$	QR
SPC2 [13]	$\mathcal{O}(\text{nnz}(A) \cdot \log n)$ + `ER_small`	$6d^2$	QR+ER
SPC3 (proposed in this article)	$\mathcal{O}(\text{nnz}(A) \cdot \log n)$ + `QR_small`	$\mathcal{O}(d^{\frac{19}{4}} \log^{\frac{11}{4}} d)$	QR+QR

finding a $\Pi \in \mathbb{R}^{r \times d}$ such that for any $x \in \mathbb{R}^d$,

$$(1/\kappa_1)\|Ax\|_1 \leq \|\Pi Ax\|_1 \leq \kappa_2 \|Ax\|_1, \tag{2.2}$$

where $r \ll n$ and is independent of n and the factors κ_1 and κ_2 here will be low-degree polynomials of d (and related to α and β of Definition 2.4). For example, Π could be the sparse Cauchy transform described in Lemma 2.5. After obtaining Π, by calculating a matrix R such that ΠAR^{-1} has orthonormal columns, the matrix AR^{-1} is a well-conditioned basis with $\kappa \leq d\sqrt{r}\kappa_1\kappa_2$. See Theorem 4.1 in [13] for more details. Here, the matrix R can be obtained by a QR factorization (or, alternately, the singular value decomposition). As the choice of Π varies, the condition number of AR^{-1}, i.e., $\kappa(AR^{-1})$, and the corresponding running time will also vary, and there is in general a trade-off among these.

For simplicity, the acronyms for these types of conditioning methods will come from the name of the corresponding transformations: SC stands for slow Cauchy transform from [16]; FC stands for fast Cauchy transform from [5]; and SPC1 (see Algorithm 1) will be the first method based on the sparse Cauchy transform (see Lemma 2.5). We will call the methods derived from this scheme QR-type methods.

- Via ellipsoid rounding (ER). Alternatively, one can compute a well-conditioned basis by applying ellipsoid rounding. This is a deterministic algorithm that computes an η-rounding of a centrally symmetric convex set $\mathcal{C} = \{x \in \mathbb{R}^d \,|\, \|Ax\|_1 \leq 1\}$. By η-rounding here we mean finding an ellipsoid $\mathcal{E} = \{x \in \mathbb{R}^d \,|\, \|Rx\|_2 \leq 1\}$, satisfying $\mathcal{E}/\eta \subseteq \mathcal{C} \subseteq \mathcal{E}$, which implies $\|Rx\|_2 \leq \|Ax\|_1 \leq \eta\|Rx\|_2$ for all $x \in \mathbb{R}^d$. With a transformation of the coordinates, it is not hard to show the following:

$$\|x\|_2 \leq \|AR^{-1}x\|_1 \leq \eta\|x\|_2. \tag{2.3}$$

From this, it is not hard to show the following inequalities:

$$\|AR^{-1}\|_1 \leq \sum_{j\in[d]} \|AR^{-1}e_j\|_1 \leq \sum_{j\in[d]} \eta\|e_j\|_2 \leq d\eta,$$
$$\|AR^{-1}x\|_1 \geq \|x\|_2 \geq \|x\|_\infty.$$

This directly leads to a well-conditioned matrix $U = AR^{-1}$ with $\kappa \leq d\eta$. Hence, the problem boils down to finding an η-rounding with η small in a reasonable time.

By Theorem 2.4.1 in [11], one can find a $(d(d+1))^{1/2}$-rounding in polynomial time. This result was used by [4] and [6]. As we mentioned in the previous section in Lemma 2.7, in [5] a new fast ellipsoid rounding algorithm was proposed. For an $n \times d$ matrix A with full rank, it takes at most $\mathcal{O}(nd^3 \log n)$ time to find a matrix R such that AR^{-1} is a well-conditioned basis with $\kappa \leq 2d^2$. We will call the methods derived from this scheme ER-type methods.

- Via combined QR+ER methods. Finally, one can construct a well-conditioned basis by combining QR-like and ER-like methods. For example, after we obtain R such that AR^{-1} is a well-conditioned basis, as described in Lemma 2.6, one can then construct a $(1 \pm \epsilon)$-distortion subspace-preserving sampling matrix S in $\mathcal{O}(\text{nnz}(A) \cdot \log n)$ time. We may view that the price we pay for obtaining S is very low in terms of running time. Since S is a sampling matrix with constant distortion factor and since the dimension of SA is independent of n, we can apply additional operations on that smaller matrix in order to obtain a better condition number, without much additional running time, in theory at least, if $n \gg \text{poly}(d)$, for some low-degree poly(d).

  Since the bottleneck for ellipsoid rounding is its running time, when compared to QR-type methods, one possibility is to apply ellipsoid rounding on SA. Since the bigger dimension of SA only depends on d, the running time for computing R via ellipsoid rounding will be acceptable if $n \gg \text{poly}(d)$. As for the condition number, for any general ℓ_1 subspace embedding Π satisfying (2.2), i.e., which preserves the ℓ_1 norm up to some factor determined by d, including S, if we apply ellipsoid rounding on ΠA, then the resulting R may still satisfy (2.3) with some η. In detail, viewing $R^{-1}x$ as a vector in $\mathbb{R}^d$, from (2.2), we have

$$(1/\kappa_2)\|\Pi AR^{-1}x\|_1 \leq \|AR^{-1}x\|_1 \leq \kappa_1 \|\Pi AR^{-1}x\|_1.$$

  In (2.3), replace A with ΠA, and combining the inequalities above, we have

$$(1/\kappa_2)\|x\|_2 \leq \|AR^{-1}x\|_1 \leq \eta\kappa_1\|x\|_2.$$

  With appropriate scaling, one can show that AR^{-1} is a well-conditioned matrix with $\kappa = d\eta\kappa_1\kappa_2$. Especially, when S has constant distortion, say, $(1 \pm 1/2)$, the condition number is preserved at sampling complexity $\mathcal{O}(d^2)$, while the running time has been reduced a lot, when compared to the vanilla ellipsoid rounding method. (See Algorithm 2 (SPC2) below for a version of this method.)

  A second possibility is to view S as a sampling matrix satisfying (2.2) with $\Pi = S$. Then, according to our discussion of the QR-type methods, if we compute the QR factorization of SA, we may expect the resulting AR^{-1} to be a well-conditioned basis with lower condition number κ. As for the running time, QR factorization on a smaller matrix will be inconsequential, in theory at least. (See Algorithm 3 (SPC3) below for a version of this method.)

In the remainder of this subsection, we will describe three related methods for computing a well-conditioned basis that we will use in our empirical evaluations. Recall that Table 1 provides a summary of these three methods and the other methods that we will use.

ALGORITHM 1. SPC1: Vanilla QR-type method with sparse Cauchy transform.

Input: $A \in \mathbb{R}^{n\times d}$ with full column rank.

Output: $R^{-1} \in \mathbb{R}^{d\times d}$ such that AR^{-1} is a well-conditioned basis with $\kappa \le \mathcal{O}(d^{\frac{13}{2}} \log^{\frac{11}{2}} d)$.

1: Construct a low-distortion embedding matrix $\Pi_1 \in \mathbb{R}^{r_1\times n}$ of $(\mathcal{A}, \|\cdot\|_1)$ via Lemma 2.5.

2: Compute $R \in \mathbb{R}^{d\times d}$ such that AR^{-1} is a well-conditioned basis for $\mathcal{A}$ via QR factorization of $\Pi_1 A$.

ALGORITHM 2. SPC2: QR + ER-type method with sparse Cauchy transform.

Input: $A \in \mathbb{R}^{n\times d}$ with full column rank.

Output: $R^{-1} \in \mathbb{R}^{d\times d}$ such that AR^{-1} is a well-conditioned basis with $\kappa \le 6d^2$.

1: Construct a low-distortion embedding matrix $\Pi_1 \in \mathbb{R}^{r_1\times n}$ of $(\mathcal{A}, \|\cdot\|_1)$ via Lemma 2.5.

2: Construct $\tilde{R} \in \mathbb{R}^{d\times d}$ such that $A\tilde{R}^{-1}$ is a well-conditioned basis for $\mathcal{A}$ via QR factorization of $\Pi_1 A$.

3: Compute a $(1 \pm 1/2)$-distortion sampling matrix $\tilde{S} \in \mathbb{R}^{\mathrm{poly}(d)\times n}$ of $(\mathcal{A}, \|\cdot\|_1)$ via Lemma 2.6.

4: Compute $R \in \mathbb{R}^{d\times d}$ by ellipsoid rounding for $\tilde{S}A$ via Lemma 2.7.

We start with the algorithm obtained when we use the sparse Cauchy transform from [13] as the random projection Π in a vanilla QR-type method. We call it SPC1 since we will describe two of its variants below. Our main result for Algorithm 1 is given in Lemma 2.9. Since the proof is quite straightforward, we omit it here.

LEMMA 2.9. *Given $A \in \mathbb{R}^{n\times d}$ with full rank, Algorithm* 1 *takes $\mathcal{O}(\mathrm{nnz}(A) \cdot \log n)$ time to compute a matrix $R \in \mathbb{R}^{d\times d}$ such that with a constant probability, AR^{-1} is a well-conditioned basis for $\mathcal{A}$ with $\kappa \le \mathcal{O}(d^{\frac{13}{2}} \log^{\frac{11}{2}} d)$.*

Next, we summarize the two combined methods described above in Algorithms 2 and 3. Since they are variants of SPC1, we call them SPC2 and SPC3, respectively. Algorithm 2 originally appeared as the first four steps of Algorithm 2 in [13]. Our main result for Algorithm 2 is given in Lemma 2.10; since the proof of this lemma is very similar to the proof of Theorem 7 in [13], we omit it here. Algorithm 3 is new to this paper. Our main result for Algorithm 3 is given in Lemma 2.11.

LEMMA 2.10. *Given $A \in \mathbb{R}^{n\times d}$ with full rank, Algorithm* 2 *takes $\mathcal{O}(\mathrm{nnz}(A)\cdot\log n)$ time to compute a matrix $R \in \mathbb{R}^{d\times d}$ such that with a constant probability, AR^{-1} is a well-conditioned basis for $\mathcal{A}$ with $\kappa \le 6d^2$.*

LEMMA 2.11. *Given $A \in \mathbb{R}^{n\times d}$ with full rank, Algorithm* 3 *takes $\mathcal{O}(\mathrm{nnz}(A)\cdot\log n)$ time to compute a matrix $R \in \mathbb{R}^{d\times d}$ such that with a constant probability, AR^{-1} is a well-conditioned basis for $\mathcal{A}$ with $\kappa \le \mathcal{O}(d^{\frac{19}{4}} \log^{\frac{11}{4}} d)$.*

Proof. By Lemma 2.5, in step 1, Π is a low-distortion embedding satisfying (2.2) with $\kappa_1\kappa_2 = \mathcal{O}(d^3 \log^3 d)$, and $r_1 = \mathcal{O}(d^5 \log^5 d)$. As a matter of fact, as we discussed in section 2.2, the resulting AR^{-1} in step 2 is a well-conditioned basis with $\kappa = \mathcal{O}(d^{\frac{13}{2}} \log^{\frac{11}{2}} d)$. In step 3, by Lemma 2.6, the sampling complexity required for obtaining a $(1 \pm 1/2)$-distortion sampling matrix is $\tilde{s} = \mathcal{O}(d^{\frac{15}{2}} \log^{\frac{11}{2}} d)$. Finally, if we view $\tilde{S}$ as a low-distortion embedding matrix with $r = \tilde{s}$ and $\kappa_2\kappa_1 = 3$, then the resulting R in step 4 will satisfy that AR^{-1} is a well-conditioned basis with $\kappa = \mathcal{O}(d^{\frac{19}{4}} \log^{\frac{11}{4}} d)$.

Algorithm 3. SPC3: QR + QR-type method with sparse Cauchy transform.

Input: $A \in \mathbb{R}^{n\times d}$ with full column rank.
Output: $R^{-1} \in \mathbb{R}^{d\times d}$ such that AR^{-1} is a well-conditioned basis with $\kappa \leq \mathcal{O}(d^{\frac{19}{4}} \log^{\frac{11}{4}} d)$.
1: Construct a low-distortion embedding matrix $\Pi_1 \in \mathbb{R}^{r_1\times n}$ of $(\mathcal{A}, \|\cdot\|_1)$ via Lemma 2.5.
2: Construct $\tilde{R} \in \mathbb{R}^{d\times d}$ such that $A\tilde{R}^{-1}$ is a well-conditioned basis for $\mathcal{A}$ via QR factorization of $\Pi_1 A$.
3: Compute a $(1 \pm 1/2)$-distortion sampling matrix $\tilde{S} \in \mathbb{R}^{\mathrm{poly}(d)\times n}$ of $(\mathcal{A}, \|\cdot\|_1)$ via Lemma 2.6.
4: Compute $R \in \mathbb{R}^{d\times d}$ via the QR factorization of $\tilde{S}A$.

For the running time, it takes $\mathcal{O}(\mathrm{nnz}(A))$ time for completing step 1. In step 2, the running time is $r_1 d^2 = \mathrm{poly}(d)$. As Lemma 2.6 points out, the running time for constructing $\tilde{S}$ in step 3 is $\mathcal{O}(\mathrm{nnz}(A) \cdot \log n)$. Since the large dimension of $\tilde{S}A$ is a low-degree polynomial of d, the QR factorization of it costs $\tilde{s}d^2 = \mathrm{poly}(d)$ time in step 4. Overall, the running time of Algorithm 3 is $\mathcal{O}(\mathrm{nnz}(A) \cdot \log n)$. ∎

Both Algorithm 2 and Algorithm 3 have additional steps (steps 3 and 4) when compared with Algorithm 1, and this leads to some improvements, at the cost of additional computation time. For example, in Algorithm 3 (SPC3), we obtain a well-conditioned basis with smaller κ when comparing to Algorithm 1 (SPC1). As for the running time, it will still be $\mathcal{O}(\mathrm{nnz}(A) \cdot \log n)$, since the additional time is for constructing sampling matrix and solving a QR factorization of a matrix whose dimensions are determined by d. Note that when we summarize these results in Table 1, we explicitly list the additional running time for SPC2 and SPC3 in order to show the trade-off between these SPC-derived methods. We will evaluate the performance of all these methods on quantile regression problems in section 4 (except for FC, since it is similar to but worse than SPC1, and ellipsoid rounding, since on the full problem it is too expensive).

Remark. For all the methods we described above, the output is *not* the well-conditioned matrix U, but instead it is the matrix R, the inverse of which transforms A into U.

Remark. As we can see in Table 1, with respect to conditioning quality, SPC2 has the lowest condition number κ, followed by SPC3 and then SPC1, which has the worst condition number. On the other hand, with respect to running time, SPC1 is the fastest, followed by SPC3 and then SPC1, which is the slowest. (The reason for this ordering of the running time is that SPC2 and SPC3 need additional steps and ellipsoid rounding takes longer running time that doing a QR decomposition.)

3. Main theoretical results. In this section, we present our main theoretical results on $(1 \pm \epsilon)$-distortion subspace-preserving embeddings and our fast randomized algorithm for quantile regression.

3.1. Main technical ingredients. In this subsection, we present the main technical ingredients underlying our main algorithm for quantile regression. We start with a result which says that if we sample sufficiently many (but still only poly(d)) rows according to an appropriately defined nonuniform importance sampling distribution (of the form given in (3.1) below), then we obtain a $(1 \pm \epsilon)$-distortion embedding matrix with respect to the loss function of quantile regression. Note that the form of this lemma is based on ideas from [6, 5].

LEMMA 3.1 (subspace-preserving sampling lemma). *Given $A \in \mathbb{R}^{n\times d}$, let $U \in \mathbb{R}^{n\times d}$ be a well-conditioned basis for $\mathcal{A}$ with condition number κ. For $s > 0$, define*

$$\hat{p}_i \geq \min\{1, s \cdot \|U_{(i)}\|_1/\|U\|_1\}, \tag{3.1}$$

and let $S \in \mathbb{R}^{n\times n}$ be a random diagonal matrix with $S_{ii} = 1/\hat{p}_i$ with probability $\hat{p}_i$ and 0 otherwise. Then when $\epsilon < 1/2$ and

$$s \geq \frac{\tau}{1-\tau}\frac{27\kappa}{\epsilon^2}\left(d\log\left(\frac{\tau}{1-\tau}\frac{18}{\epsilon}\right) + \log\left(\frac{4}{\delta}\right)\right)$$

with probability at least $1-\delta$, for every $x \in \mathbb{R}^d$,

$$(1-\varepsilon)\rho_\tau(Ax) \leq \rho_\tau(SAx) \leq (1+\varepsilon)\rho_\tau(Ax). \tag{3.2}$$

Proof. Since U is a well-conditioned basis for the range space of A, to prove (3.2) it is equivalent to prove the following holds for all $y \in \mathbb{R}^d$:

$$(1-\varepsilon)\rho_\tau(Uy) \leq \rho_\tau(SUy) \leq (1+\varepsilon)\rho_\tau(Uy). \tag{3.3}$$

To prove that (3.3) holds for any $y \in \mathbb{R}^d$, first we show that (3.3) holds for any fixed $y \in \mathbb{R}^d$; second, we apply a standard γ-net argument to show that (3.3) holds for every $y \in \mathbb{R}^d$.

Assume that U is (α,β)-conditioned with $\kappa = \alpha\beta$. For $i \in [n]$, let $v_i = U_{(i)}y$. Then $\rho_\tau(SUy) = \sum_{i\in[n]} \rho_\tau(S_{ii}v_i) = \sum_{i\in[n]} S_{ii}\rho_\tau(v_i)$ since $S_{ii} \geq 0$. Let $w_i = S_{ii}\rho_\tau(v_i) - \rho_\tau(v_i)$ be a random variable, and we have

$$w_i = \begin{cases} (\frac{1}{\hat{p}_i} - 1)\rho_\tau(v_i) & \text{with probability } \hat{p}_i; \\ -\rho_\tau(v_i) & \text{with probability } 1-\hat{p}_i. \end{cases}$$

Therefore, $\mathbf{E}[w_i] = 0, \mathbf{Var}[w_i] = (\frac{1}{\hat{p}_i} - 1)\rho_\tau(v_i)^2, |w_i| \leq \frac{1}{\hat{p}_i}\rho_\tau(v_i)$. Note here we only consider i such that $s \cdot \|U_{(i)}\|_1/\|U\|_1 < 1$ since otherwise we have $\hat{p}_i = 1$, and the corresponding term will not contribute to the variance. According to our definition, $\hat{p}_i \geq s \cdot \|U_{(i)}\|_1/\|U\|_1 = s \cdot t_i$. Consider the following:

$$\rho_\tau(v_i) = \rho_\tau(U_{(i)}y) \leq \tau\|U_{(i)}y\|_1 \leq \tau\|(U_{(i)})\|_1\|y\|_\infty.$$

Hence,

$$\begin{aligned} |w_i| &\leq \frac{1}{\hat{p}_i}\rho_\tau(v_i) \leq \frac{1}{\hat{p}_i}\tau\|U_{(i)}\|_1\|y\|_\infty \leq \frac{\tau}{s}\|U\|_1\|y\|_\infty \\ &\leq \frac{1}{s}\frac{\tau}{1-\tau}\alpha\beta\rho_\tau(Uy) := M. \end{aligned}$$

Also,

$$\sum_{i\in[n]} \mathbf{Var}[w_i] \leq \sum_{i\in[n]} \frac{1}{\hat{p}_i}\rho_\tau(v_i)^2 \leq M\rho_\tau(Uy).$$

Applying the Bernstein inequality to the zero-mean random variables w_i gives

$$\Pr\left[\left|\sum_{i\in[n]} w_i\right| > \varepsilon\right] \leq 2\exp\left(\frac{-\varepsilon^2}{2\sum_i \mathbf{Var}[w_i] + \frac{2}{3}M\epsilon}\right).$$

Since $\sum_{i\in[n]} w_i = \rho_\tau(SUy) - \rho_\tau(Uy)$, setting ε to $\varepsilon\rho_\tau(Uy)$ and plugging all the results we derive above, we have

$$\Pr\left[|\rho_\tau(SUy) - \rho_\tau(Uy)| > \varepsilon\rho_\tau(Uy)\right] \leq 2\exp\left(\frac{-\varepsilon^2\rho_\tau^2(Uy)}{2M\rho_\tau(Uy) + \frac{2\varepsilon}{3}M\rho_\tau(Uy)}\right).$$

Let's simplify the exponential term on the right-hand side of the above expression:

$$\frac{-\varepsilon^2\rho_\tau^2(Uy)}{2M\rho_\tau(Uy) + \frac{2\varepsilon}{3}M\rho_\tau(Uy)} = \frac{-s\varepsilon^2}{\alpha\beta}\frac{1-\tau}{\tau}\frac{1}{2+\frac{2\varepsilon}{3}} \leq \frac{-s\varepsilon^2}{3\alpha\beta}\frac{1-\tau}{\tau}.$$

Therefore, when $s \geq \frac{\tau}{1-\tau}\frac{27\alpha\beta}{\epsilon^2}(d\log(\frac{3}{\gamma})+\log(\frac{4}{\delta}))$, with probability at least $1-(\gamma/3)^d\delta/2$,

$$(1-\epsilon/3)\rho_\tau(Uy) \leq \rho_\tau(SUy) \leq (1+\epsilon/3)\rho_\tau(Uy), \tag{3.4}$$

where γ will be specified later.

We will show that, for all $z \in \text{range}(U)$,

$$(1-\epsilon)\rho_\tau(z) \leq \rho_\tau(Sz) \leq (1+\epsilon)\rho_\tau(z). \tag{3.5}$$

By the positive linearity of $\rho_\tau(\cdot)$, it suffices to show the bound above holds for all z with $\|z\|_1 = 1$.

Next, let $Z = \{z \in \text{range}(U) \mid \|z\|_1 \leq 1\}$ and construct a γ-net of Z, denoted by Z_γ, such that for any $z \in Z$, there exists a $z_\gamma \in Z_\gamma$ that satisfies $\|z - z_\gamma\|_1 \leq \gamma$. By [1], the number of elements in Z_γ is at most $(3/\gamma)^d$. Hence, with probability at least $1-\delta/2$, (3.4) holds for all $z_\gamma \in Z_\gamma$.

We claim that with suitable choice γ, with probability at least $1-\delta/2$, S will be a $(1\pm 2/3)$-distortion embedding matrix of $(\mathcal{A}, \|\cdot\|_1)$. To show this, first we state a similar result for $\|\cdot\|_1$ from Theorem 6 in [6] with $p=1$ as follows.

Lemma 3.2 (ℓ_1 subspace-preserving sampling lemma). *Given $A \in \mathbb{R}^{n\times d}$, let $U \in \mathbb{R}^{n\times d}$ be an (α,β)-conditioned basis for $\mathcal{A}$. For $s > 0$, define*

$$\hat{p}_i \geq \min\{1, s\cdot\|U_{(i)}\|_1/\|U\|_1\},$$

and let $S \in \mathbb{R}^{n\times n}$ be a random diagonal matrix with $S_{ii} = 1/\hat{p}_i$ with probability $\hat{p}_i$, and 0 otherwise. Then when $\epsilon < 1/2$ and

$$s \geq \frac{32\alpha\beta}{\epsilon^2}\left(d\log\left(\frac{12}{\epsilon}\right) + \log\left(\frac{2}{\delta}\right)\right)$$

with probability at least $1-\delta$, for every $x \in \mathbb{R}^d$,

$$(1-\varepsilon)\|Ax\|_1 \leq \|SAx\|_1 \leq (1+\varepsilon)\|Ax\|_1. \tag{3.6}$$

Note here we change the constraint $\epsilon \leq 1/7$ and the original theorem to $\epsilon \leq 1/2$ above. One can easily show that the result still holds with such a setting. If we set $\epsilon = 2/3$ and the failure probability to be at most $\delta/2$, the construction of S defined above satisfies conditions of Lemma 3.2 when the expected sampling complexity $s \geq \bar{s} := 72\alpha\beta\left(d\log(18) + \log\left(\frac{4}{\delta}\right)\right)$. Then our claim for S holds. Hence we only need to make sure with suitable choice of γ, we have $s \geq \bar{s}$.

For any z with $\|z\|_1 = 1$, we have

$$\begin{aligned}
|\rho_\tau(Sz) - \rho_\tau(z)| &\le |\rho_\tau(Sz) - \rho_\tau(Sz_\gamma)| + |\rho_\tau(Sz_\gamma) - \rho_\tau(z_\gamma)| + |\rho_\tau(z_\gamma) - \rho_\tau(z)| \\
&\le \tau\|S(z - z_\gamma)\|_1 + (\epsilon/3)\rho_\tau(z_\gamma) + \tau\|z_\gamma - z\|_1 \\
&\le \tau|\|S(z - z_\gamma)\|_1 - \|(z - z_\gamma)\|_1| + (\epsilon/3)\rho_\tau(z) + (\epsilon/3)\rho_\tau(z_\gamma - z) \\
&\quad + 2\tau\|z_\gamma - z\|_1 \\
&\le 2\tau/3\|z - z_\gamma\|_1 + (\epsilon/3)\rho_\tau(z) + \tau(\epsilon/3)\|z_\gamma - z\|_1 + 2\tau\|z_\gamma - z\|_1 \\
&\le (\epsilon/3)\rho_\tau(z) + \tau\gamma(2/3 + \epsilon/3 + 2) \\
&\le \left(\epsilon/3 + \frac{\tau}{1-\tau}\gamma(2/3 + \epsilon/3 + 2)\right)\rho_\tau(z) \\
&\le \epsilon\rho_\tau(z),
\end{aligned}$$

where we take $\gamma = \frac{1-\tau}{6\tau}\epsilon$, and the expected sampling size becomes

$$s = \frac{\tau}{1-\tau}\frac{27\alpha\beta}{\epsilon^2}\left(d\log\left(\frac{\tau}{1-\tau}\frac{18}{\epsilon}\right) + \log\left(\frac{4}{\delta}\right)\right).$$

When $\epsilon < 1/2$, we will have $s > \bar{s}$. Hence the claim for S holds and (3.5) holds for every $z \in \text{range}(U)$.

Since the proof is involved with two random events with failure probability at most $\delta/2$, by a simple union bound, (3.3) holds with probability at least $1 - \delta$. Our result follows since $\kappa = \alpha\beta$. ∎

Remark. It is not hard to see that for any matrix S satisfying (3.2), the rank of A is preserved.

Remark. Given such a subspace-preserving sampling matrix, it is not hard to show that by solving the subsampled problem induced by S, i.e., by solving $\min_{x\in\mathcal{C}} \rho_\tau(SAx)$, one obtains a $(1+\epsilon)/(1-\epsilon)$-approximate solution to the original problem. For more details, see the proof for Theorem 3.4.

In order to apply Lemma 3.1 to quantile regression, we need to compute the sampling probabilities in (3.1). This requires two steps: first, find a well-conditioned basis U; second, compute the ℓ_1 row norms of U. For the first step, we can apply any method described in the previous subsection. (Other methods are possible, but Algorithms 1, 2, and 3 are of particular interest due to their nearly input-sparsity running time.) We will now present an algorithm that will perform the second step of approximating the ℓ_1 row norms of U in the allotted $\mathcal{O}(\text{nnz}(A) \cdot \log n)$ time.

Suppose we have obtained R^{-1} such that AR^{-1} is a well-conditioned basis. Next, consider computing $\hat{p}_i$ from U (or from A and R^{-1}), and note that forming U explicitly is expensive both when A is dense and when A is sparse. In practice, however, we will not need to form U explicitly, and we will not need to compute the exact value of the ℓ_1-norm of each row of U. Indeed, it suffices to get estimates of $\|U_{(i)}\|_1$, in which case we can adjust the sampling complexity s to maintain a small approximation factor. Algorithm 4 provides a way to compute the estimates of the ℓ_1 norm of each row of U fast and construct the sampling matrix. The same algorithm was used in [5] except for the choice of desired sampling complexity s and we present the entire algorithm for completeness. Our main result for Algorithm 4 is presented in Proposition 3.3.

Proposition 3.3 (fast construction of $(1\pm\epsilon)$-distortion sampling matrix). *Given a matrix $A \in \mathbb{R}^{n\times d}$, and a matrix $R \in \mathbb{R}^{d\times d}$ such that AR^{-1} is a well-conditioned basis for $\mathcal{A}$ with condition number κ, Algorithm 4 takes $\mathcal{O}(\text{nnz}(A) \cdot \log n)$ time to*

ALGORITHM 4. Fast construction of $(1 \pm \epsilon)$-distortion sampling matrix of $(\mathcal{A}, \rho_\tau(\cdot))$.

Input: $A \in \mathbb{R}^{n \times d}, R \in \mathbb{R}^{d \times d}$ such that AR^{-1} is well-conditioned with condition number κ, $\epsilon \in (0, 1/2)$, $\tau \in [1/2, 1)$.
Output: Sampling matrix $S \in \mathbb{R}^{n \times n}$.
1: Let $\Pi_2 \in \mathbb{R}^{d \times r_2}$ be a matrix of independent Cauchys with $r_2 = 15 \log(40n)$.
2: Compute $R^{-1}\Pi_2$ and construct $\Lambda = AR^{-1}\Pi_2 \in \mathbb{R}^{n \times r_2}$.
3: For $i \in [n]$, compute $\lambda_i = \text{median}_{j \in [r_2]} |\Lambda_{ij}|$.
4: For $s = \frac{\tau}{1-\tau} \frac{81\kappa}{\epsilon^2}(d \log(\frac{\tau}{1-\tau} \frac{18}{\epsilon}) + \log 80)$ and $i \in [n]$, compute probabilities

$$\hat{p}_i = \min \left\{ 1, s \cdot \frac{\lambda_i}{\sum_{i \in [n]} \lambda_i} \right\}.$$

5: Let $S \in \mathbb{R}^{n \times n}$ be diagonal with independent entries

$$S_{ii} = \begin{cases} \frac{1}{\hat{p}_i} & \text{with probability } \hat{p}_i; \\ 0 & \text{with probability } 1 - \hat{p}_i. \end{cases}$$

compute a sampling matrix $S \in \mathbb{R}^{\hat{s} \times n}$ (with only one nonzero per row), such that with probability at least 0.9, S is a $(1 \pm \epsilon)$-distortion sampling matrix. That is, for all $x \in \mathbb{R}^d$,

$$(1 - \epsilon)\rho_\tau(Ax) \le \rho_\tau(SAx) \le (1 + \epsilon)\rho_\tau(Ax). \tag{3.7}$$

Further, with probability at least $1 - o(1)$, $\hat{s} = \mathcal{O}\left(\mu\kappa d \log\left(\mu/\epsilon\right)/\epsilon^2\right)$, where $\mu = \frac{\tau}{1-\tau}$.

Proof. In this lemma, slightly different from the previous notation, we will use s and $\hat{s}$ to denote the actual number of rows selected and the input parameter for defining the sampling probability, respectively. From Lemma 3.1, a $(1 \pm \epsilon)$-distortion sampling matrix S could be constructed by calculating the ℓ_1 norms of the rows of AR^{-1}. Indeed, we will estimate these row norms and adjust the sampling complexity s. According to Lemma 12 in [5], with probability at least 0.95, the $\lambda_i, i \in [n]$, we compute in the first three steps of Algorithm 4 satisfy

$$\frac{1}{2}\|U_{(i)}\|_1 \le \lambda_i \le \frac{3}{2}\|U_{(i)}\|_1,$$

where $U = AR^{-1}$. Conditioned on this high-probability event, we set

$$\hat{p}_i \ge \min \left\{ 1, \hat{s} \cdot \frac{\lambda_i}{\sum_{i \in [n]} \lambda_i} \right\}.$$

Then we will have $\hat{p}_i \ge \min\{1, \frac{\hat{s}}{3} \cdot \frac{\|U_{(i)}\|_1}{\|U\|_1}\}$. Since $\hat{s}/3$ satisfies the sampling complexity required in Lemma 3.1 with $\delta = 0.05$, the corresponding sampling matrix S is constructed as desired. These are done in steps 4 and 5. Since the algorithm involves two random events, by a simple union bound, with probability at least 0.9, S is a $(1 \pm \epsilon)$-distortion sampling matrix.

By the definition of sampling probabilities, $\mathbf{E}[s] = \sum_{i \in [n]} \hat{p}_i \le \hat{s}$. Note here s is the sum of some random variables and it is tightly concentrated around its expectation. By a standard Bernstein bound, with probability $1 - o(1)$, $s \le 2\hat{s} = \mathcal{O}\left(\mu\kappa d \log\left(\mu/\epsilon\right)/\epsilon^2\right)$, where $\mu = \frac{\tau}{1-\tau}$, as claimed.

ALGORITHM 5. Fast randomized algorithm for quantile regression.

Input: $A \in \mathbb{R}^{n\times d}$ with full column rank, $\epsilon \in (0, 1/2)$, $\tau \in [1/2, 1)$.
Output: An approximated solution $\hat{x} \in \mathbb{R}^d$ to problem $\text{minimize}_{x\in\mathcal{C}}\, \rho_\tau(Ax)$.

1: Compute $R \in \mathbb{R}^{d\times d}$ such that AR^{-1} is a well-conditioned basis for $\mathcal{A}$ via Algorithm 1, 2, or 3.
2: Compute a $(1 \pm \epsilon)$-distortion embedding $S \in \mathbb{R}^{n\times n}$ of $(\mathcal{A}, \rho_\tau(\cdot))$ via Algorithm 4.
3: Return $\hat{x} \in \mathbb{R}^d$ that minimizes $\rho_\tau(SAx)$ with respect to $x \in \mathcal{C}$.

Now let's compute the running time in Algorithm 4. The main computational cost comes from steps 2, 3, and 5. The running time in other steps will be dominated by it. It takes d^2r_2 time to compute $R^{-1}\Pi_2$; then it takes $\mathcal{O}(\text{nnz}(A)\cdot r_2)$ time to compute $AR^{-1}\Pi_2$; and finally it takes $\mathcal{O}(n)$ time to compute all the λ_i and construct S. Since $r_2 = \mathcal{O}(\log n)$, in total, the running time is $\mathcal{O}((d^2 + \text{nnz}(A))\log n + n) = \mathcal{O}(\text{nnz}(A)\cdot \log n)$. ∎

Remark. Such a technique can also be used to fast approximate the ℓ_2 row norms of a well-conditioned basis by postmultiplying a matrix consisted of Gaussian variables; see [7].

Remark. In the text before Proposition 3.3, s denotes an input parameter for defining the importance sampling probabilities. However, the actual sample size might be less than that. Since Proposition 3.3 is about the construction of the sampling matrix S, we let $\hat{s}$ denote the actual number of row selected. Also, as stated, the output of Algorithm 4 is an $n \times n$ matrix, but if we zero-out the all-zero rows, then the actual size of S is indeed $\hat{s}$ by d as described in Proposition 3.3. Throughout the following text, by sampling size s, we mean the desired sampling size which is the parameter in the algorithm.

3.2. Main algorithm. In this subsection, we state our main algorithm for computing an approximate solution to the quantile regression problem. Recall that to compute a relative-error approximate solution, it suffices to compute a $(1\pm\epsilon)$-distortion sampling matrix S. To construct S, we first compute a well-conditioned basis U with Algorithm 1, 2, or 3 (or some other conditioning method), and then we apply Algorithm 4 to approximate the ℓ_1 norm of each row of U. These procedures are summarized in Algorithm 5. The main quality-of-approximation result for this algorithm by using Algorithm 2 is stated in Theorem 3.4.

THEOREM 3.4 (fast quantile regression). *Given $A \in \mathbb{R}^{n\times d}$ and $\varepsilon \in (0, 1/2)$, if Algorithm 2 is used in step 1, Algorithm 5 returns a vector $\hat{x}$ that, with probability at least 0.8, satisfies*

$$\rho_\tau(A\hat{x}) \leq \left(\frac{1+\varepsilon}{1-\varepsilon}\right)\rho_\tau(Ax^*),$$

where x^* is an optimal solution to the original problem. In addition, the algorithm to construct $\hat{x}$ runs in time

$$\mathcal{O}(\text{nnz}(A)\cdot\log n) + \phi\left(\mathcal{O}(\mu d^3\log(\mu/\epsilon)/\epsilon^2), d\right),$$

where $\mu = \frac{\tau}{1-\tau}$ and $\phi(s, d)$ is the time to solve a quantile regression problem of size $s \times d$.

Proof. In step 1, by Lemma 2.10, the matrix $R \in \mathbb{R}^{d\times d}$ computed by Algorithm 2 satisfies that with probability at least 0.9, AR^{-1} is a well-conditioned basis for $\mathcal{A}$ with $\kappa = 6d^2$. The probability bound can be attained by setting the corresponding constants sufficiently large. In step 2, when we apply Algorithm 4 to construct the sampling matrix S, by Proposition 3.3, with probability at least 0.9, S will be a $(1 \pm \epsilon)$-distortion sampling matrix of $(\mathcal{A}, \rho_\tau(\cdot))$. Solving the subproblem $\min_{x\in\mathcal{C}} \rho_\tau(SAx)$ gives a $(1+\epsilon)/(1-\epsilon)$ solution to the original problem (1.3). This is because

$$\rho_\tau(A\hat{x}) \leq \frac{1}{1-\varepsilon}\rho_\tau(SA\hat{x}) \leq \frac{1}{1-\varepsilon}\rho_\tau(SAx^*) \leq \frac{1+\varepsilon}{1-\varepsilon}\rho_\tau(Ax^*), \tag{3.8}$$

where the first and third inequalities come from (3.7) and the second inequality comes from the fact that $\hat{x}$ is the minimizer of the subproblem. Hence the solution $\hat{x}$ returned by step 3 satisfies our claim. The whole algorithm involves two random events, and the overall success probability is at least 0.8.

Now let's compute the running time for Algorithm 5. In step 1, by Lemma 2.10, the running time for Algorithm 2 to compute R is $\mathcal{O}(\operatorname{nnz} A)$. By Proposition 3.3, the running time for step 2 is $\mathcal{O}(\operatorname{nnz}(A) \cdot \log n)$. Furthermore, as stated in Proposition 3.3 and $\kappa(AR^{-1}) = 2d^2$, with probability $1 - o(1)$, the actual sampling complexity is $\mathcal{O}\left(\mu d^3 \log\left(\mu/\epsilon\right)/\epsilon^2\right)$, where $\mu = \tau/(1-\tau)$, and it takes $\phi\left(\mathcal{O}\left(\mu d^3 \log\left(\mu/\epsilon\right)/\epsilon^2\right), d\right)$ time to solve the subproblem in step 3. This follows the overall running time of Algorithm 5 as claimed. ∎

Remark. As stated, Theorem 3.4 uses Algorithm 2 in step 3; we did this since it leads to the best known running time results in worst-case analysis, but our empirical results will indicate that due to various trade-offs the situation is more complex in practice.

Remark. Our theory provides a bound on the solution quality, as measured by the objective function of the quantile regression problem, and it does not provide bounds for the difference between the exact solution vector and the solution vector returned by our algorithm. We will, however, compute this latter quantity in our empirical evaluation.

4. Empirical evaluation on medium-scale quantile regression. In this section and the next section, we present our main empirical results. We have evaluated an implementation of Algorithm 5 using several different conditioning methods in step 1. We have considered both simulated data and real data, and we have considered both medium-sized data as well as terabyte-scale data. In this section, we will summarize our results for medium-sized data. The results on terabyte-scale data can be found in section 5.

Simulated skewed data. For the synthetic data, in order to increase the difficulty for sampling, we will add imbalanced measurements to each coordinates of the solution vector. A similar construction for the test data appeared in [5]. Due to the skewed structure of the data, we will call this data set "skewed data" in the following discussion. This data set is generated in the following way:

1. Each row of the design matrix A is a canonical vector. Suppose the number of measurements on the jth column is c_j, where $c_j = qc_{j-1}$, for $j = 2, \ldots, d$. Here $1 < q \leq 2$. A is an $n \times d$ matrix.

2. The true vector x^* with length d is a vector with independent Gaussian entries. Let $b^* = Ax^*$.

3. The noise vector ϵ is generated with independent Laplacian entries. We scale ϵ such that $\|\epsilon\|/\|b^*\| = 0.2$. The response vector is given by

$$b_i = \begin{cases} 500\epsilon_i & \text{with probability } 0.001; \\ b_i^* + \epsilon_i & \text{otherwise.} \end{cases}$$

When making the experiments, we require $c_1 \geq 161$. This implies that if we choose $s/n \geq 0.01$ and perform the uniform sampling, with probability at least 0.8, at least one row in the first block (associated with the first coordinate) will be selected, due to $1-(1-0.01)^{161} \geq 0.8$. Hence, if we choose $s \geq 0.01n$, we may expect uniform sampling to produce an acceptable estimation.

Real census data. For the real data, we consider a data set consisting of a 5% sample of the U.S. 2000 census data,[2] consisting of annual salary and related features of people who reported that they worked 40 or more weeks in the previous year and worked 35 or more hours per week. The size of the design matrix is 5×10^6 by 11.

The remainder of this section will consist of six subsections, the first five of which will show the results of experiments on the skewed data; section 4.6 will show the results on census data. In more detail, sections 4.1, 4.2, 4.3, and 4.4 will summarize the performance of the methods in terms of solution quality as the parameters s, n, d, and τ, respectively, are varied; section 4.5 will show how the running time changes as s, n, and d change.

Before showing the details, we provide a quick summary of the numerical results. We show the high quality of approximation on both objective value and solution vector by using our main algorithm, i.e., Algorithm 5, with various conditioning methods. Among all the conditioning methods, SPC2 and SPC3 show higher accuracy than other methods. They can achieve two-digit accuracy by sampling only 1% of the rows for a moderately large data set. Also, we show that using conditioning yields much higher accuracy, especially when approximating the solution vector, as we can see in Figure 1. Next, we demonstrate that the empirical results are consistent to our theory, that is, when we fix the lower dimension of the data set, d, and fix the conditioning method we use, we always achieve the same accuracy, regardless of how large the higher dimension n is, as shown in Figure 3. In Figure 5, we explore the relationship between the accuracy and the lower dimension d when n is fixed. The accuracy is monotonically decreasing as d increases. We also show that our algorithms are reliable for τ ranging from 0.05 to 0.95, as shown in Figure 6, and the magnitude of the relative error remains almost the same. As for the running time comparison, in Figures 7, 8, and 9, we show that the running time of Algorithm 5 with a different conditioning method is consistent with our theory. Moreover, SPC1 and SPC3 have a much better scalability than other methods, including the standard solver `ipm` and best previous sampling algorithm `prqfn`. For example, for $n = 1e6$ and $d = 280$, we can get at least one-digit accuracy in a reasonable time, while we can only solve problem with size $1e6$ by 180 exactly by using the standard solver in that same amount of time.

4.1. Quality of approximation when the sampling size s changes. As discussed in section 2.2, we can use one of several methods for the conditioning step, i.e., for finding the well-conditioned basis $U = AR^{-1}$ in step 1 of Algorithm 5. Here, we will consider the empirical performance of *six* methods for doing this conditioning step, namely, SC, SPC1, SPC2, SPC3, NOCO, and UNIF. The first four methods (SC, SPC1, SPC2, SPC3) are described in section 2.2; NOCO stands for "no conditioning," meaning the matrix R in the conditioning step is taken to be identity; UNIF stands

[2]U.S. Census, http://www.census.gov/census2000/PUMS5.html.

for the uniform sampling method, which we include here for completeness. Note that for all the methods, we compute the row norms of the well-conditioned basis exactly instead of estimating them with Algorithm 4. The reason is that this permits a cleaner evaluation of the quantile regression algorithm, as this may reduce the error due to the estimating step. We have, however, observed similar results if we approximate the row norms well.

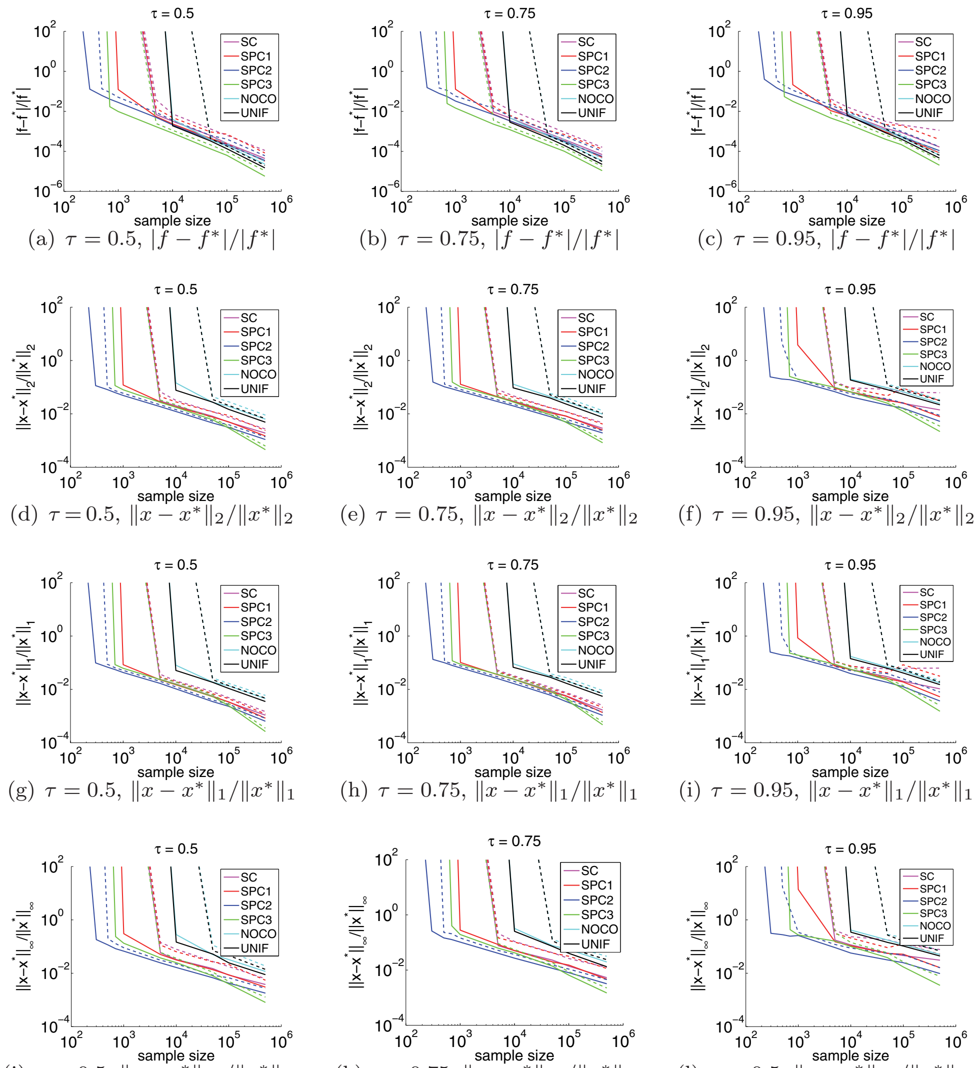
Rather than determining the sample size from a given tolerance ϵ, we let the sample size s vary in a range as an input to the algorithm. Also, for a fixed data set, we will show the results when $\tau = 0.5, 0.75, 0.95$. In our figure, we will plot the first and the third quartiles of the relative errors of the objective value and solution measured in three different norms from 50 independent trials. We restrict the y axis in the plots to the range of $[0, 100]$ to show more details. We start with a test on skewed data with size $1e6 \times 50$. (Recall that by $1e6 \times 50$, we mean that $n = 1 \times 10^6$ and $d = 50$.) The resulting plots are shown in Figure 1.

From these plots, if we look at the sampling size required for generating at least one-digit accuracy, then SPC2 needs the fewest samples, followed by SPC3 and then SPC1. This is consistent with the order of the condition numbers of these methods. For SC, although in theory it has good condition number properties, in practice it performs worse than other methods. Not surprisingly, NOCO and UNIF are not reliable when s is very small, e.g., less than $1e4$.

When the sampling size s is large enough, the accuracy of each conditioning method is close to the others in terms of the objective value. Among these, SPC3 performs slightly better than others. When estimating the actual solution vectors, the conditioning-based methods behave substantially better than the two naive methods. SPC2 and SPC3 are the most reliable methods since they can yield the least relative error for every sample size s. NOCO is likely to sample the outliers, and UNIF cannot get accurate answer until the sampling size $s \geq 1e4$. This accords with our expectations. For example, when s is less than $1e4$, as we pointed out in the remark below the description of the skewed data, it is very likely that none of the rows in the first block corresponding to the first coordinate will be selected. Thus, poor estimation will be generated due to the imbalanced measurements in the design matrix. Note that from the plots we can see that if a method fails with some sampling complexity s, then for that value of s the relative errors will be huge (e.g., larger than 100, which is clearly a trivial result). Note also that all the methods can generate at least one-digit accuracy if s is large enough.

It is worth mentioning the performance difference among SPC1, SPC2, and SPC3. In Table 1, we show the trade-off between running time and condition number for the three methods. As we pointed out, SPC2 always needs the least sampling complexity to generate two-digit accuracy, followed by SPC3 and then SPC1. When s is large enough, SPC2 and SPC3 perform substantially better than SPC1. As for the running time, SPC1 is the fastest, followed by SPC3 and then SPC2. Again, all these follow the theory about our SPC methods. We will present a more detailed discussion for the running time in section 4.5.

Although our theory doesn't say anything about the quality of the solution vector itself (as opposed to the value of the objective function), we evaluate this here. To measure the approximation to the solution vectors, we use three norms (the ℓ_1, ℓ_2, and ℓ_∞ norms). From Figure 1, we see that the performance among these methods is qualitatively similar for each of the three norms, but the relative error is higher when measured in the ℓ_∞ norm. For more detail, see Table 2, where we show the exact



(a) $\tau = 0.5$, $|f - f^*|/|f^*|$ (b) $\tau = 0.75$, $|f - f^*|/|f^*|$ (c) $\tau = 0.95$, $|f - f^*|/|f^*|$

(d) $\tau = 0.5$, $\|x - x^*\|_2/\|x^*\|_2$ (e) $\tau = 0.75$, $\|x - x^*\|_2/\|x^*\|_2$ (f) $\tau = 0.95$, $\|x - x^*\|_2/\|x^*\|_2$

(g) $\tau = 0.5$, $\|x - x^*\|_1/\|x^*\|_1$ (h) $\tau = 0.75$, $\|x - x^*\|_1/\|x^*\|_1$ (i) $\tau = 0.95$, $\|x - x^*\|_1/\|x^*\|_1$

(j) $\tau = 0.5$, $\|x - x^*\|_\infty/\|x^*\|_\infty$ (k) $\tau = 0.75$, $\|x - x^*\|_\infty/\|x^*\|_\infty$ (l) $\tau = 0.5$, $\|x - x^*\|_\infty/\|x^*\|_\infty$

Fig. 1. *The first (solid lines) and third (dashed lines) quartiles of the relative errors of the objective value (namely,* $|f - f^*|/|f^*|$*) and solution vector (measured in three different norms, namely, the* $\ell_2, \ell_1,$ *and* ℓ_∞ *norms), by using* 6 *different methods, among* 50 *independent trials. The test is on skewed data with size* 1e6 *by* 50*. The three columns correspond to* $\tau = 0.5, 0.75, 0.95$*, respectively.*

quartiles of the relative error on vectors for each methods for $s = 5e4$ and $\tau = 0.75$. Not surprisingly, NOCO and UNIF are not among the reliable methods when s is small (and they get worse when s is even smaller). Note that the relative error for each method doesn't change substantially when τ takes different values. We present a more detailed discussion of the τ dependence in section 4.4.

(We note also that for subsequent figures in subsequent subsections, we obtained similar qualitative trends for the errors in the approximate solution vectors when the errors were measured in different norms. Thus, due to this similarity and to save space, in subsequent figures we will only show errors for the ℓ_2 norm.)

Table 2

The first and third quartiles of relative errors of the solution vector, measured in ℓ_1, ℓ_2, and ℓ_∞ norms. The test data set is the skewed data with size 1e6 $\times$ 50*, the sampling size $s = 5e4$, and $\tau = 0.75$.*

	$\|x - x^*\|_2/\|x^*\|_2$	$\|x - x^*\|_1/\|x^*\|_1$	$\|x - x^*\|_\infty/\|x^*\|_\infty$
SC	[0.0121, 0.0172]	[0.0093, 0.0122]	[0.0229, 0.0426]
SPC1	[0.0108, 0.0170]	[0.0081, 0.0107]	[0.0198, 0.0415]
SPC2	[0.0079, 0.0093]	[0.0061, 0.0071]	[0.0115, 0.0152]
SPC3	[0.0094, 0.0116]	[0.0086, 0.0103]	[0.0139, 0.0184]
NOCO	[0.0447, 0.0583]	[0.0315, 0.0386]	[0.0769, 0.1313]
UNIF	[0.0396, 0.0520]	[0.0287, 0.0334]	[0.0723, 0.1138]

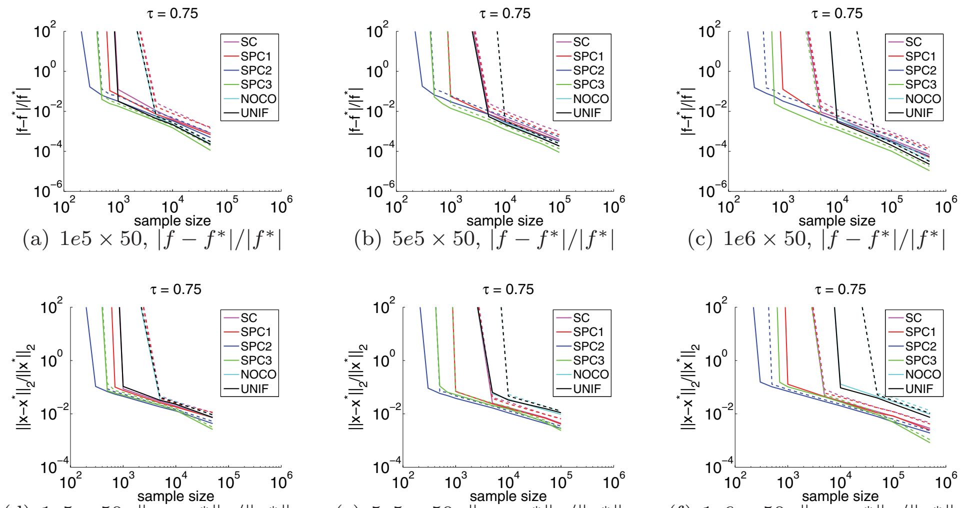
4.2. Quality of approximation when the higher dimension n changes. Next, we describe how the performance of our algorithm varies when higher dimension n changes. (We present the results when the lower dimension d changes in section 4.3.) Figures 2 and 3 summarize our results.

Figure 2 shows the performance of the relative error of the objective value and solution vector by using the six different methods, as n is varied, for fixed values of $\tau = 0.75$ and $d = 50$. For each row, the three figures come from three data sets with n taking value in $1e5, 5e5, 1e6$. (Recall that in these experiments, we only list the plots showing the relative error on vectors measured in ℓ_2 norm. Since the plots for the ℓ_1 and ℓ_∞ norms are similar, we omit them.) We see that when d is fixed, the basic structure in the plots that we observed before is preserved when n takes three different values. In particular, the minimum sampling complexity s needed for each method for yielding high accuracy does not vary a lot. When s is large enough, the relative performance among all the methods is similar, and when all the parameters are fixed except for n, the relative error for each method does not change quantitatively.

We will also let n take a wider range of values. Figure 3 shows the change of relative error on the objective value and solution vector by using SPC3 and letting n vary from $1e4$ to $1e6$ and $d = 50$ fixed. Recall from Theorem 3.4 that for a given tolerance ϵ, the required sampling complexity s depends only on d. That is, if we fix the sampling size s and d, then the relative error should not vary much, as a function of n. If we inspect Figure 3, we see that the relative errors are almost constant as a function of increasing n, provided that n is much larger than s. When s is very close to n, since we are sampling roughly the same number of rows as in the full data, we should expect lower errors. Also, we can see that by using SPC3, relative errors remain roughly the same in magnitude.

4.3. Quality of approximation when the lower dimension d changes. Next, we describe how the overall performance changes when the lower dimension d changes. Figures 4 and 5 summarize our results. These figures show the same quantities that were plotted in the previous subsection, except that here it is the lower dimension d that is now changing, and the higher dimension $n = 1e6$ is fixed. In Figure 4, we let d take values in $10, 50, 100$, we set $\tau = 0.75$, and we show the relative error for all six conditioning methods. In Figure 5, we let d take more values in the range of $[10, 100]$, and we show the relative errors by using SPC3 for different sampling sizes s and τ values.

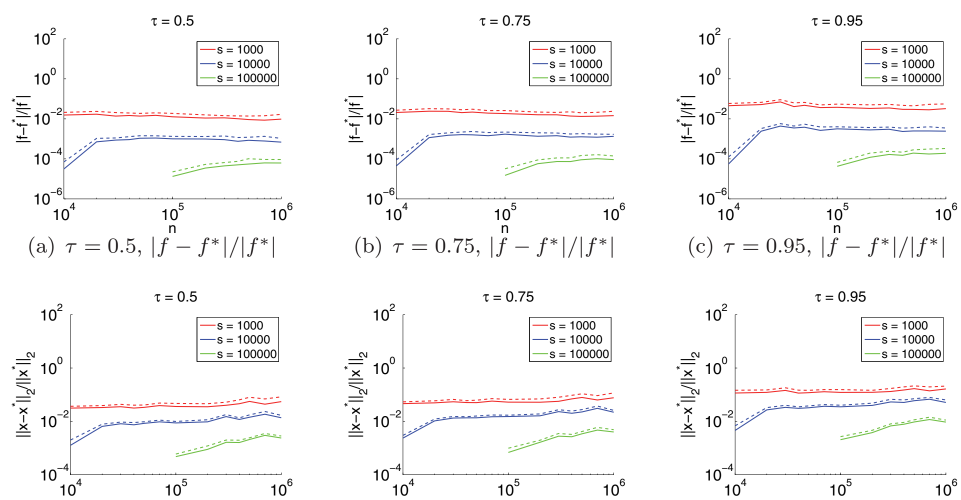
For Figure 4, as d gets larger, the performance of the two naive methods does not vary a lot. However, this increases the difficulty for conditioning methods to yield two-digit accuracy. When d is quite small, most methods can yield two-digit accuracy



(a) $1e5 \times 50$, $|f - f^*|/|f^*|$ (b) $5e5 \times 50$, $|f - f^*|/|f^*|$ (c) $1e6 \times 50$, $|f - f^*|/|f^*|$

(d) $1e5 \times 50$, $\|x - x^*\|_2/\|x^*\|_2$ (e) $5e5 \times 50$, $\|x - x^*\|_2/\|x^*\|_2$ (f) $1e6 \times 50$, $\|x - x^*\|_2/\|x^*\|_2$

Fig. 2. *The first (solid lines) and third (dashed lines) quartiles of the relative errors of the objective value (namely,* $|f - f^*|/|f^*|$*) and solution vector (namely,* $\|x - x^*\|_2/\|x^*\|_2$*), when the sample size* s *changes, for different values of* n*, while* $d = 50$ *by using* 6 *different methods, among* 50 *independent trials. The test is on skewed data and* $\tau = 0.75$*. The three columns correspond to* $n = 1e5, 5e5, 1e6$*, respectively.*

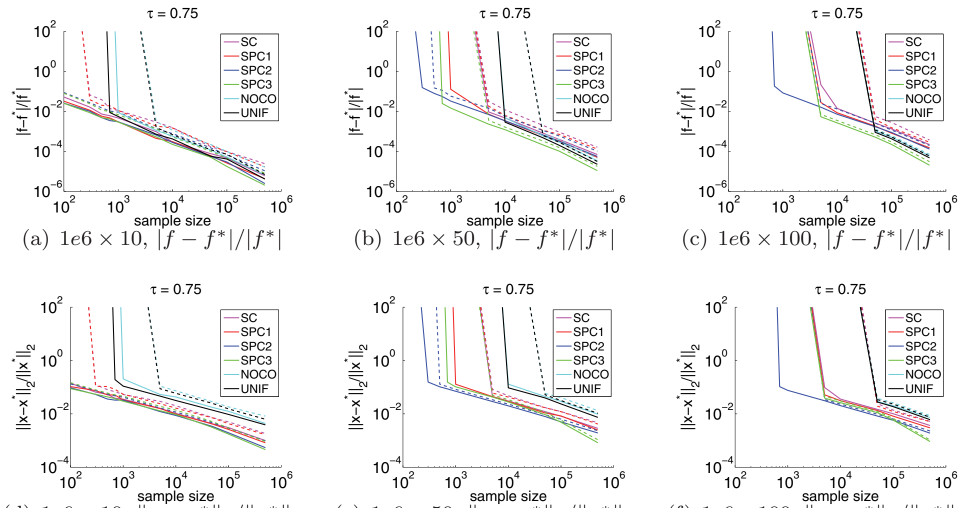


(a) $\tau = 0.5$, $|f - f^*|/|f^*|$ (b) $\tau = 0.75$, $|f - f^*|/|f^*|$ (c) $\tau = 0.95$, $|f - f^*|/|f^*|$

(d) $\tau = 0.5$, $\|x - x^*\|_2/\|x^*\|_2$ (e) $\tau = 0.75$, $\|x - x^*\|_2/\|x^*\|_2$ (f) $\tau = 0.95$, $\|x - x^*\|_2/\|x^*\|_2$

Fig. 3. *The first (solid lines) and third (dashed lines) quartiles of the relative errors of the objective value (namely,* $|f - f^*|/|f^*|$*) and solution vector (namely,* $\|x - x^*\|_2/\|x^*\|_2$*), when* n *varying from* 1e4 *to* 1e6 *and* $d = 50$ *by using SPC*3*, among* 50 *independent trials. The test is on skewed data. The three columns correspond to* $\tau = 0.5, 0.75, 0.95$*, respectively.*

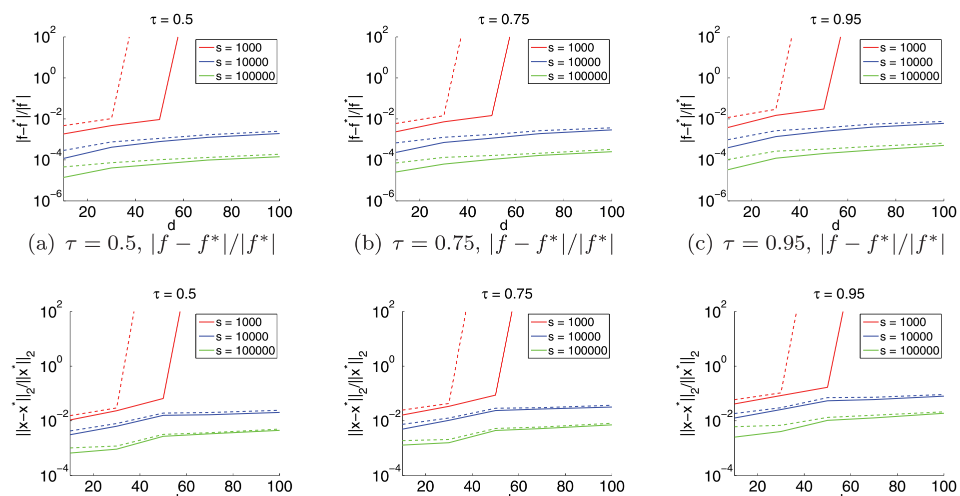
even when s is not large. When d becomes large, SPC2 and SPC3 provide good estimation, even when $s < 1000$. The relative performance among these methods remains unchanged. For Figure 5, the relative errors are monotonically increasing for each sampling size. This is consistent with our theory that to yield high accuracy, the required sampling size is a low-degree polynomial of d.



(a) $1e6 \times 10$, $|f - f^*|/|f^*|$ (b) $1e6 \times 50$, $|f - f^*|/|f^*|$ (c) $1e6 \times 100$, $|f - f^*|/|f^*|$

(d) $1e6 \times 10$, $\|x - x^*\|_2/\|x^*\|_2$ (e) $1e6 \times 50$, $\|x - x^*\|_2/\|x^*\|_2$ (f) $1e6 \times 100$, $\|x - x^*\|_2/\|x^*\|_2$

FIG. 4. *The first (solid lines) and third (dashed lines) quartiles of the relative errors of the objective value (namely, $|f - f^*|/|f^*|$) and solution vector (namely, $\|x - x^*\|_2/\|x^*\|_2$), when the sample size s changes, for different values of d, while $n = 1e6$ by using 6 different methods, among 50 independent trials. The test is on skewed data and $\tau = 0.75$. The three columns correspond to $d = 10, 50, 100$, respectively.*

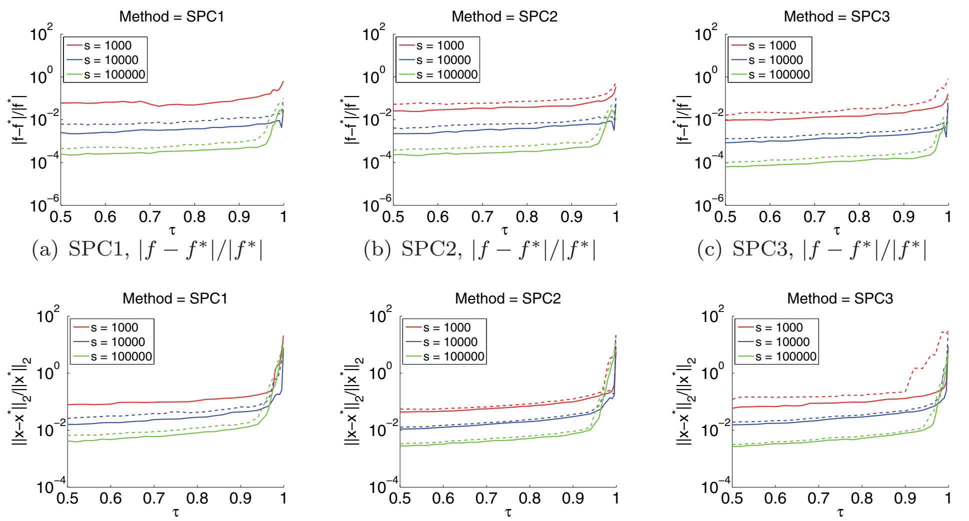


(a) $\tau = 0.5$, $|f - f^*|/|f^*|$ (b) $\tau = 0.75$, $|f - f^*|/|f^*|$ (c) $\tau = 0.95$, $|f - f^*|/|f^*|$

(d) $\tau = 0.5$, $\|x - x^*\|_2/\|x^*\|_2$ (e) $\tau = 0.75$, $\|x - x^*\|_2/\|x^*\|_2$ (f) $\tau = 0.95$, $\|x - x^*\|_2/\|x^*\|_2$

FIG. 5. *The first (solid lines) and the third (dashed lines) quartiles of the relative errors of the objective value (namely, $|f - f^*|/|f^*|$) and solution vector (namely, $\|x - x^*\|_2/\|x^*\|_2$), when d varying from 10 to 100 and $n = 1e6$ by using SPC3, among 50 independent trials The test is on skewed data. The three columns correspond to $\tau = 0.5, 0.75, 0.95$, respectively.*

4.4. Quality of approximation when the quantile parameter τ changes. Next, we will let τ change, for a fixed data set and fixed conditioning method, and we will investigate how the resulting errors behave as a function of τ. We will consider τ in the range of $[0.5, 0.9]$, equally spaced by 0.05, as well as several extreme quantiles



(a) SPC1, $|f - f^*|/|f^*|$ (b) SPC2, $|f - f^*|/|f^*|$ (c) SPC3, $|f - f^*|/|f^*|$

(d) SPC1, $\|x - x^*\|_2/\|x^*\|_2$ (e) SPC2, $\|x - x^*\|_2/\|x^*\|_2$ (f) SPC3, $\|x - x^*\|_2/\|x^*\|_2$

FIG. 6. *The first (solid lines) and the third (dashed lines) quartiles of the relative errors of the objective value (namely,* $|f - f^*|/|f^*|$*) and solution vector (namely,* $\|x - x^*\|_2/\|x^*\|_2$*), when* τ *varies from* 0.5 *to* 0.999 *by using SPC*1*, SCP*2*, SPC*3*, among* 50 *independent trials. The test is on skewed data with size* 1e6 *by* 50*. Within each plot, three sampling sizes are considered, namely,* 1e4, 1e4, 1e5*.*

such as 0.975 and 0.98. We consider skewed data with size $1e6 \times 50$; our plots are shown in Figure 6.

The plots in Figure 6 demonstrate that given the same method and sampling size s, the relative errors are monotonically increasing but only very gradually, i.e., they do not change very substantially in the range of $[0.5, 0.95]$. On the other hand, all the methods generate high relative errors when τ takes extreme values very near 1 (or 0). Overall, SPC2 and SPC3 perform better than SPC1. Although for some quantiles SPC3 can yield slightly lower errors than SPC2, it too yields worst results when τ takes on extreme values.

4.5. Evaluation on running time performance. In this subsection, we will describe running time issues, with an emphasis on how the running time behaves as a function of s, d, and n.

When the sampling size s changes. To start, Figure 7 shows the running time for computing three subproblems associated with three different τ values by using six methods (namely, SC, SPC1, SPC2, SPC3, NOCO, UNIF) when the sampling size s changes. (This is simply the running time comparison for all six methods used to generate Figure 1.) As expected, the two naive methods (NOCO and UNIF) run faster than other methods in most cases—since they don't perform the additional step of conditioning. For $s < 10^4$, among the conditioning-based methods, SPC1 runs fastest, followed by SPC3 and then SPC2. As s increases, however, the faster methods, including NOCO and UNIF, become relatively more expensive, and when $s \approx 5e5$, all the curves, except for SPC1, reach almost the same point.

To understand what is happening here, recall that we accept the sampling size s as an input in our algorithm; we then construct our sampling probabilities by $\hat{p}_i = \min\{1, s \cdot \lambda_i / \sum \lambda_i\}$, where λ_i is the estimation of the ℓ_1 norm of the ith row of a well-conditioned basis. (See step 4 in Algorithm 4.) Hence, s is not the exact

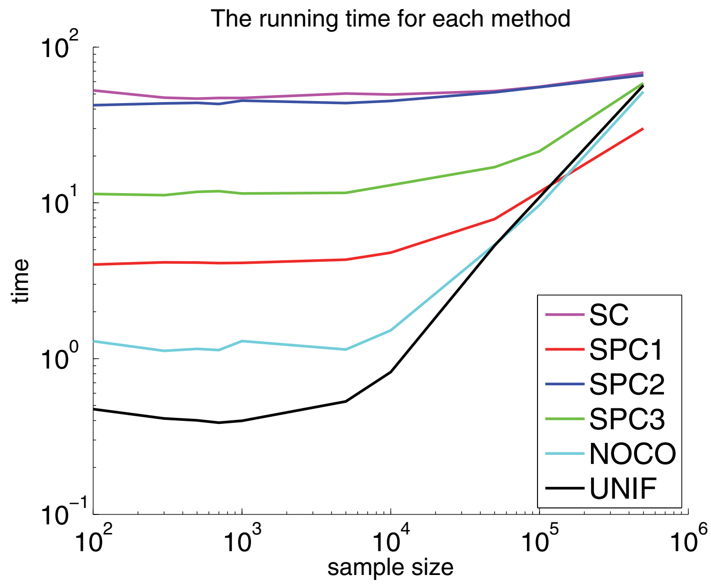


FIG. 7. *The running time for solving the three problems associated with three different τ values by using six methods, namely, SC, SPC1, SPC2, SPC3, NOCO, UNIF, when the sampling size s changes.*

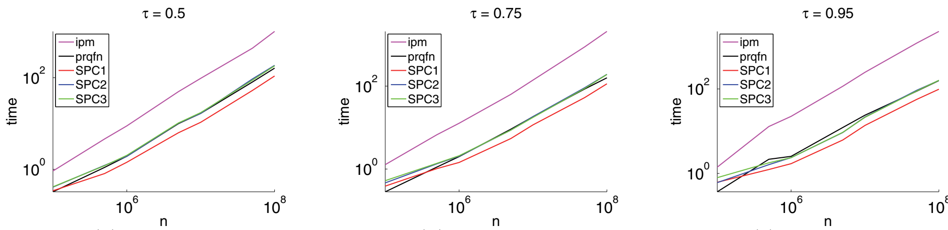
sampling size. Indeed, upon examination, in this regime when s is large, the actual sampling size is often much less than the input s. As a result, almost all the conditioning-based algorithms are solving a subproblem with size, say, $s/2 \times d$, while the two naive methods are solving subproblem with size about $s \times d$. The difference of running time for solving problems with these sizes can be quite large when s is large. For conditioning-based algorithms, the running time mainly comes from the time for conditioning and solving the subproblem. Thus, since SPC1 needs the least time for conditioning, it should be clear why SPC1 needs much less time when s is very large.

When the higher dimension n changes. Next, we compare the running time of our method with some competing methods when data size increases. The competing methods are the primal-dual method, referred to as `ipm`, and that with preprocessing, referred to as `prqfn`; see [15] for more details on these two methods.

We let the large dimension n increase from $1e5$ to $1e8$, and we fix $s = 5e4$. For completeness, in addition to the skewed data, we will consider two additional data sets. First, we also consider a design matrix with entries generated from independently and identically distributed Gaussian distribution, where the response vector is generated in the same manner as the skewed data. Also, we will replicate the census data 20 times to obtain a data set with size $1e8$ by 11. For each n, we extract the leading $n \times d$ submatrix of the replicated matrix, and we record the corresponding running time. The results of running time on all three data sets are shown in Figure 8.

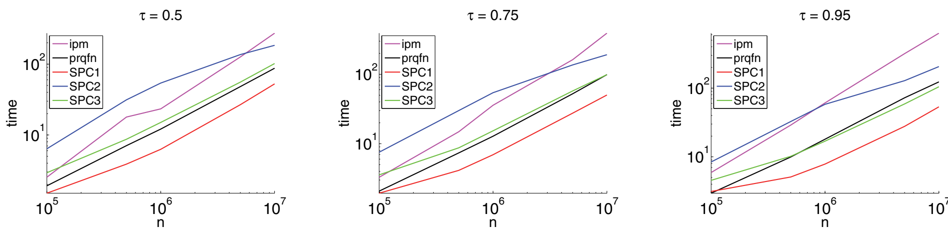
From the plots in Figure 8 we see that SPC1 runs faster than any other methods across all the data sets, in some cases significantly so. SPC2, SPC3, and `prqfn` perform similarly in most cases, and they appear to have a linear rate of increase. Also, the relative performance between each method does not vary a lot as the data type changes.

Notice that for the skewed data, when $d = 50$, SPC2 runs much slower than when $d = 10$. The reason for this is that for conditioning-based methods, the running time



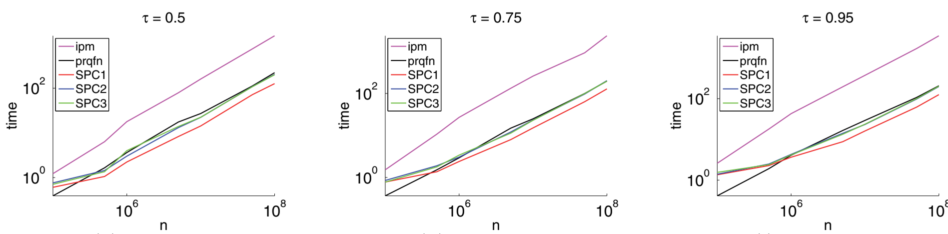
(a) $\tau = 0.5$ (b) $\tau = 0.75$ (c) $\tau = 0.95$

Skewed data with $d = 11$



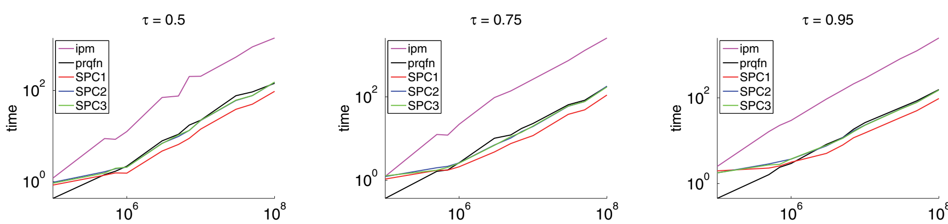
(d) $\tau = 0.5$ (e) $\tau = 0.75$ (f) $\tau = 0.95$

Skewed data with $d = 50$



(g) $\tau = 0.5$ (h) $\tau = 0.75$ (i) $\tau = 0.95$

Gaussian data with $d = 11$

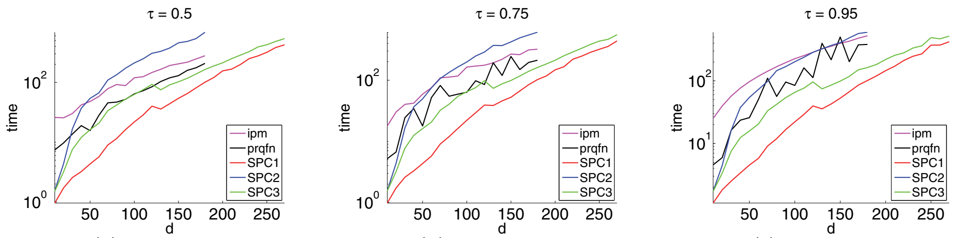


(j) $\tau = 0.5$ (k) $\tau = 0.75$ (l) $\tau = 0.95$

Replicated census data with $d = 11$

FIG. 8. *The running time for five methods (*`ipm`*,* `prqfn`*, SPC1, SPC2, and SPC3) on the same data set with d fixed and n changing. The sampling size $s = 5e4$, and the three columns correspond to $\tau = 0.5, 0.75, 0.95$, respectively.*

is composed of two parts, namely, the time for conditioning and the time for solving the subproblem. For SPC2, an ellipsoid rounding needs to be applied on a smaller data set whose larger dimension is a polynomial of d. When the sampling size s is small, i.e., the size of the subproblem is not too large, the dominant running time for SPC2 will be the time for ellipsoid rounding, and as d increase (by, say, a factor of 5) we expect a worse running time. Notice also that for all the methods, the running time does not vary a lot when τ changes. Finally, notice that all the conditioning-based methods run faster on skewed data, especially when d is small. The reason is that the



(a) $\tau = 0.5$ (b) $\tau = 0.75$ (c) $\tau = 0.95$

FIG. 9. *The running time for five methods (`ipm`, `prqfn`, SPC1, SPC2, and SPC3) for solving skewed data, with $n = 1e6$, $s = 1e4$, when d varies. SPC1 and SPC3 show better scaling than other methods when $d < 180$. For this reason, we keep running the experiments for SPC1 and SPC3 until $d = 270$. When $d < 100$, the three conditioning-based methods can yield two-digit accuracy. When for $d \in [100, 180]$, they can yield one-digit accuracy.*

running time for these three methods is of the order of input-sparsity time, and the skewed data are very sparse.

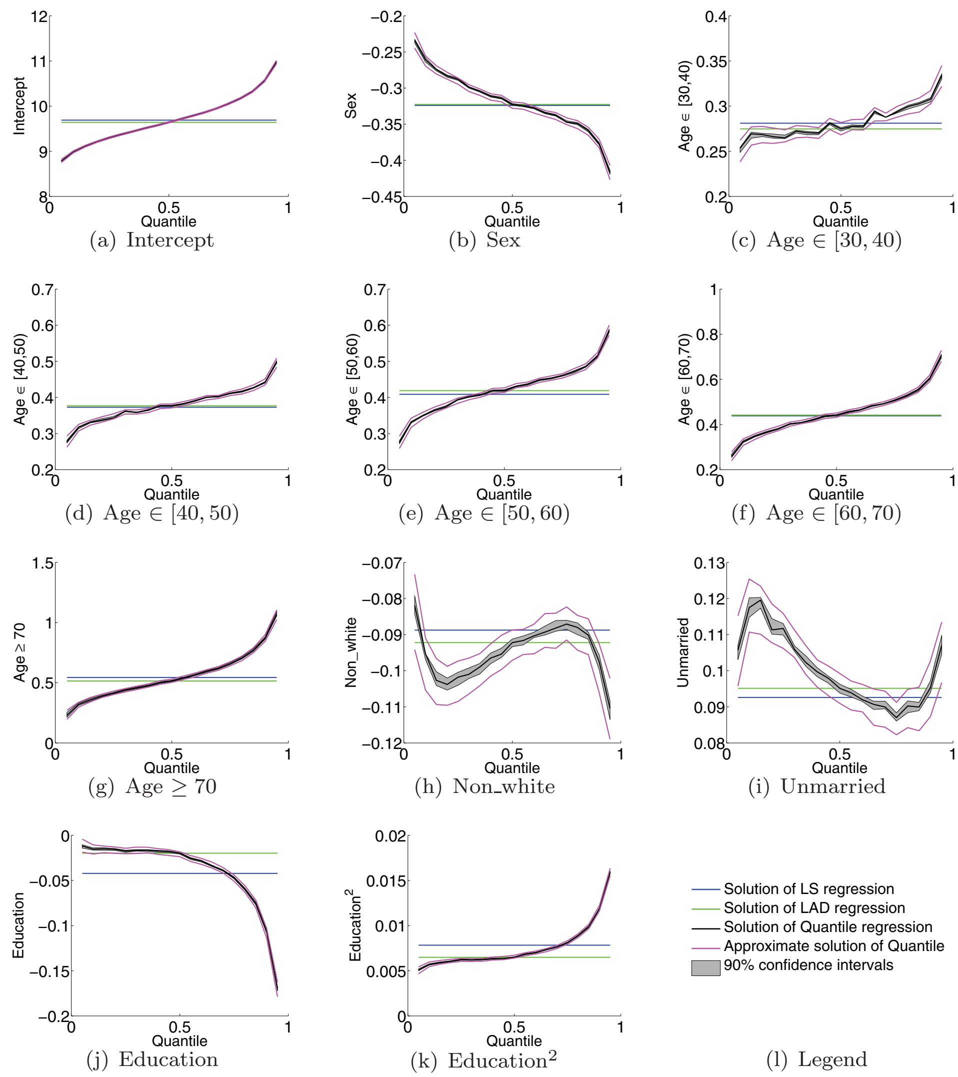
When the lower dimension d changes. Finally, we will describe the scaling of the running time as the lower dimension d changes. To do so, we fixed $n = 1e6$ and the sampling size $s = 1e4$. We let all five methods run on the data set with d varying from 5 up to 180. When $d \approx 200$, the scaling was such that all the methods except for SPC1 and SPC3 became too expensive. Thus, we let only SPC1 and SPC3 run on additional data sets with d up to 270. The plots are shown in Figure 9.

From the plots in Figure 9, we can see that when $d < 180$, SPC1 runs significantly faster than any other method, followed by SPC3 and `prqfn`. The performance of `prqfn` is quite variable. The reason for this is that there is a step in `prqfn` that involves uniform sampling, and the number of subproblems to be solved in each time might vary a lot. The scalings of SPC2 and `ipm` are similar, and when d gets much larger, say, $d > 200$, they may not be favorable due to the running time. When $d < 180$, all the conditioning methods can yield at least one-digit accuracy. Although one can only get an approximation to the true solution by using SPC1 and SPC3, they will be a good choice when d gets even larger, say, up to several hundred, as shown in Figure 9. We note that we could let d get even larger for SPC1 and SPC3, demonstrating that SPC1 and SPC3 are able to run with a much larger lower dimension than the other methods.

Remark. One may notice a slight but sudden change in the running time for SPC1 and SPC3 at $d \approx 130$. After we traced down the reason, we found that the difference come from the time in the conditioning step (since the subproblems they are solving have similar size), especially the time for performing the QR factorization. At this size, it will be normal to take more time to factorize a slightly smaller matrix due to the structure of the cache line, and it is for this reason that we see that minor decrease in running time with increasing d. We point out that the running time of our conditioning-based algorithm is mainly affected by the time for the conditioning step. That is also the reason why it does not vary a lot when τ changes.

4.6. Evaluation on solution of census data. Here, we will describe more about the accuracy of the census data when SPC3 is applied to it. The size of the census data is roughly $5e6 \times 11$.

We will generate plots that are similar to those in [10]. For each coefficient, we will compute a few quantities of it, as a function of τ, when τ varies from 0.05 to



(a) Intercept (b) Sex (c) Age $\in [30, 40)$

(d) Age $\in [40, 50)$ (e) Age $\in [50, 60)$ (f) Age $\in [60, 70)$

(g) Age ≥ 70 (h) Non_white (i) Unmarried

(j) Education (k) Education^2 (l) Legend

FIG. 10. *Each subfigure is associated with a coefficient in the census data. The shaded area shows a pointwise* 90% *confidence interval. The black curve inside is the true solution when* τ *changes from* 0.05 *to* 0.95. *The blue and green lines correspond to the* ℓ_2 *and* ℓ_1 *solutions, respectively. The two magenta curves show the first and third quartiles of solutions obtained by using SPC*3*, among* 200 *independent trials with sampling size* $s = 5e4$ *(about* 1% *of the original data).*

0.95. We compute a pointwise 90% confidence interval for each τ by bootstrapping. These are shown as the shaded area in each subfigure. Also, we compute the quartiles of the approximated solutions by using SPC3 from 200 independent trials with sampling size $s = 5e4$ to show how close we can get to the confidence interval. In addition, we also show the solution to the least square regression (LS) and least absolute deviations regression (LAD) on the same problem. The plots are shown in Figure 10.

From these plots we can see that although the two quartiles are not inside the confidence interval, they are quite close, even for this value of s. The sampling size in each trial is only $5e4$, which is about 1% of the original data, while for bootstrapping, we are resampling the same number of rows as in the original matrix with replacement. In addition, the median of these 50 solutions is in the shaded area and close to the true solution. Indeed, for most of the coefficients, SPC3 can generate two-digit accuracy. Note that we also computed the exact values of the quartiles; we don't present them here since they are very similar to those in Table 4 in terms of accuracy. See Table 4 in section 5 for more details. All in all, SPC3 performs quite well on this real data.

5. Empirical evaluation on large-scale quantile regression. In this section, we continue our empirical evaluation with an evaluation of our main algorithm applied to terabyte-scale problems. Here, the data sets are generated by "stacking" the medium-scale data a few thousand times. Although this leads to "redundant" data, which may favor sampling methods, this has the advantage that it leads to terabyte-sized problems whose optimal solution at different quantiles are known. At this terabyte scale, `ipm` has two major issues: memory requirement and running time. Although shared memory machines with more than a terabyte RAM exist, they are rare in practice (in 2013). Instead, the MapReduce framework is the de facto standard parallel environment for large data analysis. Apache Hadoop,[3] an open source implementation of MapReduce, is widely used in practice. Since our sampling algorithm only needs several passes through the data and it is embarrassingly parallel, it is straightforward to implement it on Hadoop.

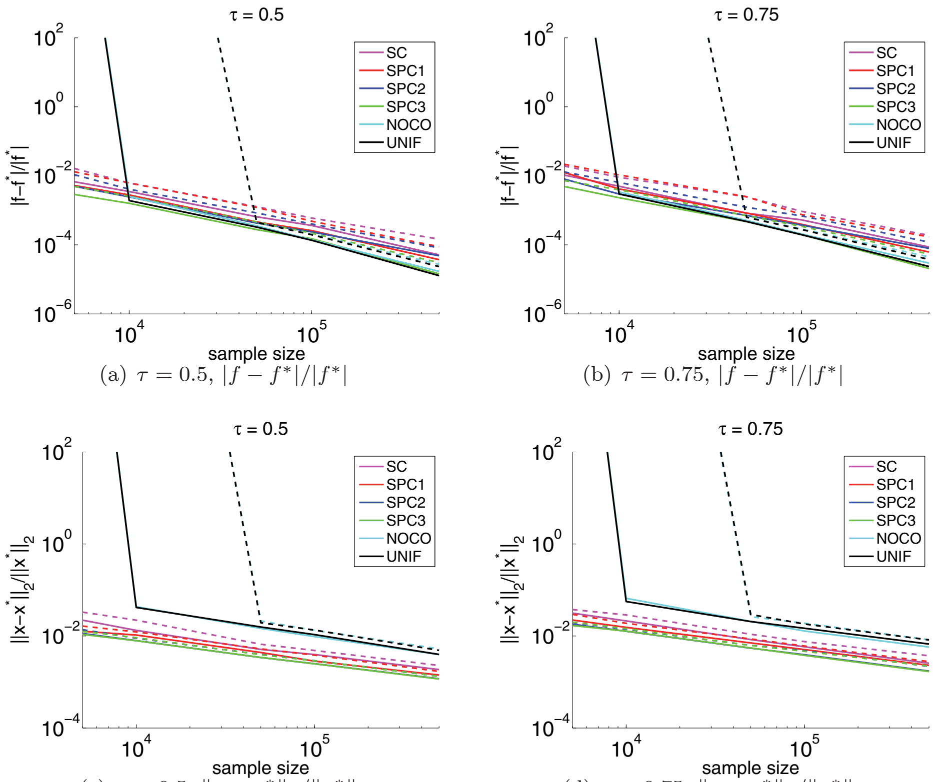
For skewed data with size $1e6 \times 50$, we stack it vertically 2500 times. This leads to data with size $2.5e9 \times 50$. In order to show the evaluations similar to Figure 1, we still implement SC, SPC1, SPC2, SPC3, NOCO, and UNIF. Figure 11 shows the relative errors on the replicated skewed data set by using the six methods. We only show the results for $\tau = 0.5$ and 0.75 since the conditioning methods tend to generate abnormal results when $\tau = 0.95$. These plots correspond with and should be compared to the four subfigures in the first two rows and columns of Figure 1.

As can be seen, the method preserves the same structure as when the method is applied to the medium-scale data. Still, SPC2 and SPC3 perform slightly better than other methods when s is large enough. In this case, as before, NOCO and UNIF are not reliable when $s < 1e4$. When $s > 1e4$, NOCO and UNIF perform sufficiently closely to the conditioning-based methods on approximating the objective value. However, the gap between the performance on approximating the solution vector is significant.

In order to show more detail on the quartiles of the relative errors, we generated a table similar to Table 2 which records the quartiles of relative errors on vectors, measured in ℓ_1, ℓ_2, and ℓ_∞ norms by using the six methods when the sampling size $s = 5e4$ and $\tau = 0.75$. Table 3 shows similar quantities to and should be compared with Table 2. Conditioning-based methods can yield two-digit accuracy when $s = 5e4$, while NOCO and UNIF cannot. Also, the relative error is somewhat higher when measured in the ℓ_∞ norm.

Next, we will explore how the accuracy may change as the lower dimension d varies and explore the capacity of our large-scale version algorithm. In this experiment, we fix the higher dimension of the replicated skewed data to be $1e9$ and let d take values in $10, 50, 100, 150$. We will only use SPC2 as it has the relative best condition number. Figure 12 shows the results of the experiment described above.

[3]Apache Hadoop, http://hadoop.apache.org/.



(a) $\tau = 0.5$, $|f - f^*|/|f^*|$

(b) $\tau = 0.75$, $|f - f^*|/|f^*|$

(c) $\tau = 0.5$, $\|x - x^*\|_2/\|x^*\|_2$

(d) $\tau = 0.75$, $\|x - x^*\|_2/\|x^*\|_2$

Fig. 11. *The first (solid lines) and the third (dashed lines) quartiles of the relative errors of the objective value (namely, $|f - f^*|/|f^*|$) and solution vector (namely, $\|x - x^*\|_2/\|x^*\|_2$), by using 6 different methods, among 30 independent trials, as a function of the sample size s. The test is on replicated skewed data with size 2.5e9 by 50. The two columns correspond to $\tau = 0.5, 0.75$, respectively.*

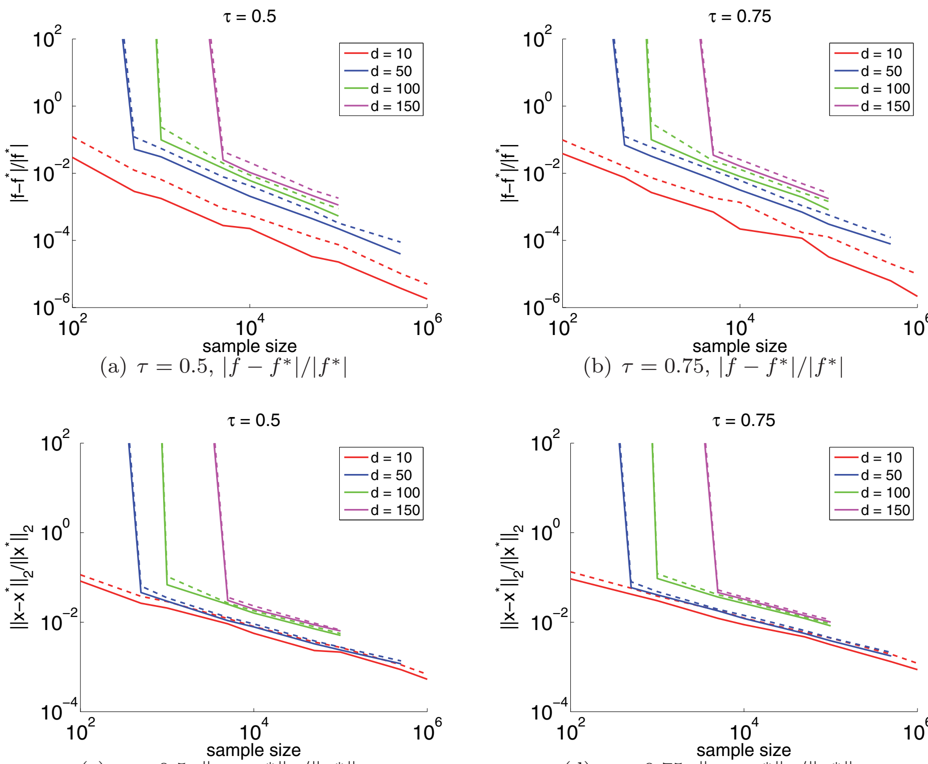
Table 3

The first and the third quartiles of relative errors of the solution vector, measured in ℓ_1, ℓ_2, and ℓ_∞ norms. The test is on replicated synthetic data with size 2.5e9 by 50, the sampling size $s = 5e4$, and $\tau = 0.75$.

	$\|x - x^*\|_2/\|x^*\|_2$	$\|x - x^*\|_1/\|x^*\|_1$	$\|x - x^*\|_\infty/\|x^*\|_\infty$
SC	[0.0084, 0.0109]	[0.0075, 0.0086]	[0.0112, 0.0159]
SPC1	[0.0071, 0.0086]	[0.0066, 0.0079]	[0.0080, 0.0105]
SPC2	[0.0054, 0.0063]	[0.0053, 0.0061]	[0.0050, 0.0064]
SPC3	[0.0055, 0.0062]	[0.0054, 0.0064]	[0.0050, 0.0067]
NOCO	[0.0207, 0.0262]	[0.0163, 0.0193]	[0.0288, 0.0397]
UNIF	[0.0206, 0.0293]	[0.0175, 0.0200]	[0.0242, 0.0474]

From Figure 12, except for some obvious facts such as the accuracies become lower as d increases when the sampling size is unchanged, we should also notice that the lower d is, the higher the minimum sampling size required to yield acceptable relative errors will be. For example, when $d = 150$, we need to sample at least $1e4$ rows in order to obtain at least one-digit accuracy.

Notice also that there are some missing points in the plot. That means we cannot solve the subproblem at that sampling size with certain d. For example, solving a



(a) $\tau = 0.5$, $|f - f^*|/|f^*|$ (b) $\tau = 0.75$, $|f - f^*|/|f^*|$

(c) $\tau = 0.5$, $\|x - x^*\|_2/\|x^*\|_2$ (d) $\tau = 0.75$, $\|x - x^*\|_2/\|x^*\|_2$

Fig. 12. *The first (solid lines) and the third (dashed lines) quartiles of the relative errors of the objective value (namely, $|f - f^*|/|f^*|$) and solution vector (namely, $\|x - x^*\|_2/\|x^*\|_2$), by using SPC2, among 30 independent trials, as a function of the sample size s. The test is on replicated skewed data with $n = 1e9$ and $d = 10, 50, 100, 150$. The two columns correspond to $\tau = 0.5, 0.75$, respectively. The missing points mean that the subproblem on such sampling size with corresponding d is unsolvable in RAM.*

subproblem with size $1e6$ by 100 is unrealistic on a single machine. Therefore, the corresponding point is missing. Another difficulty we encounter is the capability of conditioning on a single machine. Recall that in Algorithm 2, we need to perform QR factorization or ellipsoid rounding on a matrix, say, SA, whose size is determined by d. In our large-scale version of the algorithm, since these two procedures are not parallelizable, we have to perform these locally. When $d = 150$, the higher dimension of SA will be over $1e7$. Such size has reached the limit of RAM for performing QR factorization or ellipsoid rounding. Hence, it prevents us from increasing the lower dimension d.

For the census data, we stack it vertically 2000 times to construct a realistic data set whose size is roughly $1e10 \times 11$. In Table 4, we present the solution computed by our randomized algorithm with a sample size $1e5$ at different quantiles, along with the corresponding optimal solution. As can be seen, for most coefficients, our algorithm provides at least two-digit accuracy. Moreover, in applications such as this, the quantile regression result reveals some interesting facts about these data. For example, for these data, marriage may entail a higher salary in lower quantiles; Education^2, whose value ranged from 0 to 256, has a strong impact on the total income, especially in the

TABLE 4

Quantile regression results for the U.S. Census 2000 *data. The response is the total annual income. Except for the intercept and the terms involved with education, all the covariates are* $\{0, 1\}$ *binary indicators.*

Covariate	$\tau = 0.1$	$\tau = 0.25$	$\tau = 0.5$	$\tau = 0.75$	$\tau = 0.9$
intercept	8.9812 [8.9673, 8.9953]	9.3022 [9.2876, 9.3106]	9.6395 [9.6337, 9.6484]	10.0515 [10.0400, 10.0644]	10.5510 [10.5296, 10.5825]
female	-0.2609 [-0.2657, -0.2549]	-0.2879 [-0.2924, -0.2846]	-0.3227 [-0.3262, -0.3185]	-0.3472 [-0.3481, -0.3403]	-0.3774 [-0.3792, -0.3708]
Age $\in$ [30, 40)	0.2693 [0.2610, 0.2743]	0.2649 [0.2613, 0.2723]	0.2748 [0.2689, 0.2789]	0.2936 [0.2903, 0.2981]	0.3077 [0.3027, 0.3141]
Age $\in$ [40, 50)	0.3173 [0.3083, 0.3218]	0.3431 [0.3407, 0.3561]	0.3769 [0.3720, 0.3821]	0.4118 [0.4066, 0.4162]	0.4416 [0.4386, 0.4496]
Age $\in$ [50, 60)	0.3316 [0.3190, 0.3400]	0.3743 [0.3686, 0.3839]	0.4188 [0.4118, 0.4266]	0.4612 [0.4540, 0.4636]	0.5145 [0.5071, 0.5230]
Age $\in$ [60, 70)	0.3237 [0.3038, 0.3387]	0.3798 [0.3755, 0.3946]	0.4418 [0.4329, 0.4497]	0.5072 [0.4956, 0.5162]	0.6027 [0.5840, 0.6176]
Age $\geq$ 70	0.3206 [0.2962, 0.3455]	0.4132 [0.4012, 0.4359]	0.5152 [0.5036, 0.5308]	0.6577 [0.6371, 0.6799]	0.8699 [0.8385, 0.8996]
non_white	-0.0953 [-0.1023, -0.0944]	-0.1018 [-0.1061, -0.0975]	-0.0922 [-0.0985, -0.0902]	-0.0871 [-0.0932, -0.0860]	-0.0975 [-0.1041, -0.0932]
married	0.1175 [0.1121, 0.1238]	0.1117 [0.1059, 0.1162]	0.0951 [0.0918, 0.0989]	0.0870 [0.0835, 0.0914]	0.0953 [0.0909, 0.0987]
education	-0.0152 [-0.0179, -0.0117]	-0.0175 [-0.0200, -0.0149]	-0.0198 [-0.0225, -0.0189]	-0.0470 [-0.0500, -0.0448]	-0.1062 [-0.1112, -0.1032]
education^2	0.0057 [0.0055, 0.0058]	0.0062 [0.0061, 0.0064]	0.0065 [0.0064, 0.0066]	0.0081 [0.0080, 0.0083]	0.0119 [0.0117, 0.0122]

higher quantiles; the difference in age doesn't affect the total income much in lower quantiles but becomes a significant factor in higher quantiles.

To summarize our large-scale evaluation, our main algorithm can handle terabyte-sized quantile regression problems easily, obtaining, e.g., two digits of accuracy by sampling about $1e5$ rows on a problem of size $1e10 \times 11$. In addition, its running time is competitive with the best existing random sampling algorithms, and it can be applied in parallel and distributed environments. However, its capability is restricted by the size of RAM since some steps of the algorithms need to be performed locally.

6. Conclusion. We have proposed, analyzed, and evaluated new randomized algorithms for solving medium-scale and large-scale quantile regression problems. Our main algorithm uses a subsampling technique that involves constructing an ℓ_1-well-conditioned basis, and our main algorithm runs in nearly input-sparsity time, plus the time needed for solving a subsampled problem whose size depends only on the lower dimension of the design matrix. The sampling probabilities used by our main algorithm are derived by calculating the ℓ_1 norms of a well-conditioned basis, and this conditioning step is an essential step of our method. For completeness, we have provided a summary of recently proposed ℓ_1 conditioning methods, and based on this we have introduced a new method (SPC3) in this article.

We have also provided a detailed empirical evaluation of our main algorithm. This evaluation includes a comparison in terms of the quality of approximation of several variants of our main algorithm that are obtained by applying several different conditioning methods. The empirical results meet our expectation according to the theory. Most of the conditioning methods, like our proposed method, SPC3, yield two-digit accuracy by sampling only 0.1% of the data on our test problem. As for running time, our algorithm is more scalable when compared to existing competing algorithms, especially when the lower dimension gets up to several hundred, while the large dimension is at least 1 million. In addition, we show that our algorithm works well for terabyte-sized data in terms of accuracy and solvability.

Finally, we should emphasize that our main algorithm relies heavily on the notion of ℓ_1 conditioning and that the overall performance of it can be improved if better ℓ_1 conditioning methods are derived.

Acknowledgment. This work was supported in part by a grant from the Army Research Office.

REFERENCES

[1] J. Bourgain, J. Lindenstrauss, and V. Milman, *Approximation of zonoids by zonotopes*, Acta Math., 162 (1989), pp. 73–141.

[2] M. Buchinsky, *Changes is US wage structure* 1963–87*: An application of quantile regression*, Econometrica, 62 (1994), pp. 405–408.

[3] I. S. Buhai, *Quantile regression: Overview and selected applications*, Ad Astra, 4 (2005), pp. 1–17.

[4] K. Clarkson, *Subgradient and sampling algorithms for ℓ_1 regression*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 257–266.

[5] K. L. Clarkson, P. Drineas, M. Magdon-Ismail, M. W. Mahoney, X. Meng, and D. P. Woodruff, *The Fast Cauchy Transform and faster robust linear regression*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, 2013.

[6] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney, *Sampling algorithms and corsets for ℓ_p regression*, SIAM J. Comput., 38 (2009), pp. 2060–2078.

[7] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, *Fast approximation of matrix coherence and statistical leverage*, J. Mach. Learn. Res., 13 (2012), pp. 3441–3472.

[8] R. Koenker and G. Bassett, *Regression quantiles*, Econometrica, 46 (1978), pp. 33–50.

[9] R. Koenker and V. D'Orey, *Computing regression quantiles*, J. Roy. Statist. Soc. Ser. C, 43 (1993), pp. 410–414.

[10] R. Koenker and K. Hallock, *Quantile regression*, J. Econom. Perspectives, 15 (2001), pp. 143–156.

[11] L. Lovász, *Algorithmic Theory of Numbers, Graphs, and Convexity*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 50, SIAM, Philadelphia, 1986.

[12] M. W. Mahoney, *Randomized algorithms for matrices and data*, in Foundations and Trends in Machine Learning, NOW Publishers, Boston, 2011.

[13] X. Meng and M. W. Mahoney, *Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression*, in Proceedings of the 45th Annual ACM Symposium on Theory of Computing, 2013.

[14] S. Portnoy, *On computation of regression quantiles: Making the Laplacian tortoise faster*, in L_1-Statistical Procedures and Related Topics, Lecture Notes–Monograph Series 31, Institute of Mathematical Statistics, Hayward, CA, 1997, pp. 187–200.

[15] S. Portnoy and R. Koenker, *The Gaussian hare and the Laplacian tortoise: Computability of squared-error versus absolute-error estimators, with discussion*, Statist. Sci., 12 (1997), pp. 279–300.

[16] C. Sohler and D. P. Woodruff, *Subspace embedding for the ℓ_1-norm with applications*, in Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, 2011, pp. 755–764.

[17] D. P. Woodruff and Q. Zhang, *Subspace embeddings and ℓ_p-regression using exponential random variables*, in Proceedings of the 26th Conference on Learning Theory, 2013.

[18] J. Yang, X. Meng, and M. W. Mahoney, *Quantile regression for large-scale applications*, Proceedings of the 30th International Conference on Machine Learning, (2013) arXiv:abs/1305.0087.