The Fast Cauchy Transform and Faster Robust Linear Regression

Kenneth L. Clarkson [*] Petros Drineas [†] Malik Magdon-Ismail [‡]
Michael W. Mahoney [§] Xiangrui Meng [¶] David P. Woodruff [‖]

Abstract

We provide fast algorithms for overconstrained ℓ_p regression and related problems: for an $n \times d$ input matrix A and vector $b \in \mathbb{R}^n$, in $O(nd \log n)$ time we reduce the problem $\min_{x\in\mathbb{R}^d} \|Ax-b\|_p$ to the same problem with input matrix $\tilde{A}$ of dimension $s \times d$ and corresponding $\tilde{b}$ of dimension $s \times 1$. Here, $\tilde{A}$ and $\tilde{b}$ are a *coreset* for the problem, consisting of sampled and rescaled rows of A and b; and s is independent of n and polynomial in d. Our results improve on the best previous algorithms when $n \gg d$, for all $p \in [1,\infty)$ except $p = 2$; in particular, they improve the $O(nd^{1.376+})$ running time of Sohler and Woodruff (STOC, 2011) for $p = 1$, that uses asymptotically fast matrix multiplication, and the $O(nd^5 \log n)$ time of Dasgupta *et al.* (SICOMP, 2009) for general p, that uses ellipsoidal rounding. We also provide a suite of improved results for finding well-conditioned bases via ellipsoidal rounding, illustrating tradeoffs between running time and conditioning quality, including a one-pass conditioning algorithm for general ℓ_p problems.

To complement this theory, we provide a detailed empirical evaluation of implementations of our algorithms for $p = 1$, comparing them with several related algorithms. Among other things, our empirical results clearly show that, in the asymptotic regime, the theory is a very good guide to the practical performance of these algorithms. Our algorithms use our faster constructions of well-conditioned bases for ℓ_p spaces and, for $p = 1$, a fast subspace embedding of independent interest that we call the Fast Cauchy Transform: a matrix $\Pi : \mathbb{R}^n \mapsto \mathbb{R}^{O(d \log d)}$, found obliviously to A, that approximately preserves the ℓ_1 norms: that is, $\|Ax\|_1 \approx \|\Pi Ax\|_1$, for all x, with distortion $O(d^{2+\eta} \log d)$, for an arbitrarily small constant $\eta > 0$; and, moreover, ΠA can be computed in $O(nd \log d)$ time. The techniques underlying our Fast Cauchy Transform include fast Johnson-Lindenstrauss transforms, low-coherence matrices, and rescaling by Cauchy random variables.

1 Introduction

Random sampling, random projection, and other embedding methods have proven to be very useful in recent years in the development of improved worst-case algorithms for a range of linear algebra problems. For example, Gaussian random projections provide low-distortion subspace embeddings in the ℓ_2 norm, mapping an arbitrary d-dimensional subspace in $\mathbb{R}^n$ into a d-dimensional subspace in $\mathbb{R}^r$, with $r = O(d)$, and distorting the ℓ_2 norm of each vector in the subspace by at most a constant factor. Importantly for many applications, the embedding is oblivious in the sense that it is implemented by a linear mapping chosen from a distribution on mappings that is independent of the input subspace. Such low-distortion embeddings can be used to speed up various geometric algorithms, if they can be computed sufficiently quickly. As an example, the *Fast Johnson Lindenstrauss transform* (FJLT) is one such embedding; the FJLT is computable in $O(n \log d)$ time, using a variant of the fast Hadamard transform [1]. Among other things, use of the FJLT leads to faster algorithms for constructing orthonormal bases, ℓ_2 regression, and ℓ_2 subspace approximation, which in turn lead to faster algorithms for a range of related problems including low-rank matrix approximation [7, 10, 6].

In this paper, we use ℓ_1 and ℓ_p extensions of these methods to provide faster algorithms for the classical ℓ_p regression problem and several other related problems. Recall the overconstrained ℓ_p regression problem.

Definition 1. *Given a matrix $A \in \mathbb{R}^{n\times d}$, with $n > d$, a vector $b \in \mathbb{R}^n$, and a norm $\|\cdot\|_p$, the ℓ_p* regression problem *is to find the optimal solution to:*

$$(1.1) \qquad \mathcal{Z} = \min_{x\in\mathbb{R}^d} \|Ax-b\|_p.$$

[*]IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120. Email: klclarks@us.ibm.com

[†]Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180. Email: drinep@cs.rpi.edu

[‡]Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180. Email: magdon@cs.rpi.edu

[§]Dept. of Mathematics, Stanford University, Stanford, CA 94305. Email: mmahoney@cs.stanford.edu

[¶]ICME, Stanford University, Stanford, CA 94305. Email: mengxr@stanford.edu

[‖]IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120. Email: dpwoodru@us.ibm.com

In this paper, we are most interested in the case $p = 1$, although many of our results hold more generally, and so we state several of our results for general p. The ℓ_1 regression problem, also known as the Least Absolute Deviations or Least Absolute Errors problem, is especially of interest as a more robust alternative to the ℓ_2 regression or Least Squares Approximation problem.

It is well-known that for $p \geq 1$, the ℓ_p regression problem is a convex optimization problem; and for $p = 1$ and $p = \infty$, it is an instance of linear programming. Recent work has focused on using sampling, projection, and other embedding methods to solve these problems more quickly than with general convex programming or linear programming methods. Most relevant for our work is the work of Clarkson [3] on solving the ℓ_1 regression problem with subgradient and sampling methods; the work of Dasgupta *et al.* [5] on using well-conditioned bases and subspace-preserving sampling algorithms to solve general ℓ_p regression problems; and the work of Sohler and Woodruff [13] on using the Cauchy Transform to obtain improved ℓ_1 embeddings, thereby leading to improved algorithms for the ℓ_1 regression problem. The Cauchy Transform of [13] provides low-distortion embeddings for the ℓ_1 norm, and thus it is an ℓ_1 analog of the Gaussian projection for ℓ_2. It consists of a dense matrix of Cauchy random variables, and so it is "slow" to apply to an arbitrary matrix A; but since it provides the first analog of the Johnson-Lindenstrauss embedding for the ℓ_1 norm, it can be used to speed up randomized algorithms for problems such as ℓ_1 regression and ℓ_1 subspace approximation [13].

In this paper, we provide fast algorithms for overconstrained ℓ_p regression and several related problems. Our algorithms use our faster constructions of well-conditioned bases for ℓ_p spaces; and, for $p = 1$, our algorithms use a fast subspace embedding of independent interest that we call the Fast Cauchy Transform (FCT). We also provide a detailed empirical evaluation of the FCT and its use at computing ℓ_1 well-conditioned bases and solving ℓ_1 regression problems.

The FCT is our main technical result, and it is essentially an ℓ_1 analog of the FJLT. The FCT can be represented by a matrix $\Pi : \mathbb{R}^n \mapsto \mathbb{R}^{O(d \log d)}$, found obliviously to A (in the sense that its construction does not depend on any information in A), that approximately preserves the ℓ_1 norms of all vectors in $\{Ax \mid x \in \mathbb{R}^d\}$. That is, $\|Ax\|_1 \approx \|\Pi Ax\|_1$, for all x, with distortion $O(d^{2+\eta} \log d)$, for an arbitrarily small constant $\eta > 0$ (see Theorem 3.2); and, moreover, ΠA can be computed in $O(nd \log d)$ time. We actually provide two related constructions of the FCT (see Theorems 3.1 and 3.2). The techniques underlying our FCTs include FJLTs, low-coherence matrices, and rescaling by Cauchy random variables.

Our main application of the FCT embedding is to constructing the current fastest algorithm for computing a *well-conditioned basis* for ℓ_1 (see Theorem 4.1). Such a basis is an analog for the ℓ_1 norm of what an orthonormal basis is for the ℓ_2 norm, and our result improves the result in [13]. We also provide a generalization of this result to constructing ℓ_p well-conditioned bases (see Theorem 5.3). The main application for well-conditioned bases is to regression: if the rows of A are sampled according to probabilities derived from the norms of the rows of such a basis, the resulting sample of rows (and corresponding entries of b) are with high probability a *coreset* for the regression problem; see, *e.g.*, [5]. That is, for an $n \times d$ input matrix A and vector $b \in \mathbb{R}^n$, we can reduce an ℓ_p regression problem to another ℓ_p regression problem with input matrix $\tilde{A}$ of dimension $s \times d$ and corresponding $\tilde{b}$ of dimension $s \times 1$. Here, $\tilde{A}$ and $\tilde{b}$ consist of sampled and rescaled rows of A and b; and s is independent of n and polynomial in d. We point out that our construction uses as a black box an FJLT, which means that any improvement in the running time of the FJLT (for example exploiting the sparsity of A) results in a corresponding improvement to the running times of our ℓ_p regression.

Based on our constructions of well-conditioned bases, we give the fastest known construction of coresets for ℓ_p regression, for all $p \in [1, \infty)$, except $p = 2$. In particular, for ℓ_1 regression, we construct a coreset of size $\frac{1}{\varepsilon^2} \operatorname{poly}(d, \log \frac{1}{\varepsilon})$ that achieves a $(1+\varepsilon)$-approximation guarantee (see Theorem 4.2). Our construction runs in $O(nd \log n)$ time, improving the previous best algorithm of Sohler and Woodruff [13], which has an $O(nd^{1.376+})$ running time. Our extension to finding an ℓ_p well-conditioned basis also leads to an $O(nd \log n)$ time algorithm for a $(1+\varepsilon)$-approximation to the ℓ_p regression problem (see Theorem 5.4), improving the $O(nd^5 \log n)$ algorithm of Dasgupta *et al.* [5]. For $p = 1$, extensions of our basic methods yield improved algorithms for several related problems. For example, we can further optimize the running time for $p = 1$ to $O(nd \log(\varepsilon^{-1} d \log n))$; we can generalize our ℓ_1 result to solving the multiple regression problem; and we can use this to give the current fastest algorithm for computing a $(1+\varepsilon)$-approximation for the ℓ_1 subspace approximation problem.

In addition to our construction of ℓ_p well-conditioned bases (see Theorem 5.3) and their use in providing a $(1 + \varepsilon)$-approximation to the ℓ_p regression problem (see Theorem 5.4), we also provide a suite of improved results for finding well-conditioned bases via ellipsoidal rounding for general ℓ_p problems, illustrating tradeoffs between running time and conditioning quality. These methods complement the FCT-based methods in the sense that the FCT may be viewed as a tool to compute a good basis in an oblivious manner, and the ellipsoid-based methods provide an alternate way to compute a good basis in a data-dependent manner. In particular, we prove that we can obtain an ellipsoidal rounding matrix in at most $O(nd^3 \log n)$ time that provides a $2d$-rounding (see Theorem 5.2). This is much faster than the algorithm of Lovász [9] that computes a $(d(d+1))^{1/2}$-rounding in $O(nd^5 \log n)$ time. We also present an optimized algorithm that uses an FJLT to compute a well-conditioned basis of A in $O(nd \log n)$ time (see Theorem 5.3). When $p = 1$, these ℓ_p rounding algorithms are competitive with or better than previous algorithms that were developed for ℓ_1.

Finally, we also provide the first empirical evaluation for this class of randomized algorithms. In particular, we provide a detailed evaluation of a numerical implementation of both FCT constructions, and we compare the results with an implementation of the (slow) Cauchy Transform, as well as a Gaussian Transform and an FJLT. These latter two are ℓ_2-based projections. We evaluate the quality of the ℓ_1 well-conditioned basis, the core component in all our geometric algorithms, on a suite of matrices designed to test the limits of these randomized algorithms, and we also evaluate how the method performs in the context of ℓ_1 regression. This latter evaluation includes an implementation on a nearly terabyte-scale problem, where we achieve a 10^{-3} relative-error approximation to the optimal solution, a task that was infeasible prior to our work. Among other things, our empirical results clearly show that, in the asymptotic regime, the theory is a very good guide to the practical performance of these algorithms.

We provide here a brief outline of this paper. We start in Section 2 with some preliminaries, including several technical results that we will use in our analysis and that are of independent interest. Then, in Section 3, we will present our main technical results for the Fast Cauchy Transform; and in Section 4, we will describe applications of it to ℓ_1 well-conditioned basis construction and ℓ_1 leverage score approximation, to solving the ℓ_1 regression problem, and to solving the ℓ_1 norm subspace approximation problem. Then, in Section 5, we describe extensions of these ideas to general ℓ_p problems. Section 6 will contain a detailed empirical evaluation of our algorithms for ℓ_1-based problems, including the construction of ℓ_1 well-conditioned bases and both small-scale and large-scale ℓ_1 regression problems. Section 7 will then contain a brief conclusion. Proofs of all of our results, as well as many additional results, including a more detailed empirical evaluation, can be found in the technical report version of this conference paper [4].

2 Preliminaries

Let $A \in \mathbb{R}^{n\times d}$ be an $n \times d$ input matrix, where we assume $n \gg d$ and A has full column rank. The task of *linear regression* is to find the vector $x^* \in \mathbb{R}^d$ that minimizes $\|Ax - b\|$ with respect to x, for a given $b \in \mathbb{R}^n$ and norm $\|\cdot\|$. In this paper, our focus is mostly on the ℓ_1 norm, although we also discuss extensions to ℓ_p, for any $p \geq 1$. Recall that, for $p \in [1, \infty]$, the ℓ_p norm of a vector x is $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$, defined to be $\max_i |x_i|$ for $p = \infty$. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$; and let $A_{(i)}$ and $A^{(j)}$ be the ith row vector and jth column vector of A, respectively. For matrices, we use the Frobenius norm $\|A\|_F^2 = \sum_{i=1}^n \sum_{j=1}^d A_{ij}^2$, the ℓ_2-operator (or spectral) norm $\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2$, and the entrywise ℓ_p norm $\|X\|_p = (\sum_{i,j} |X_{ij}|^p)^{1/p}$. (The exception to this is $p = 2$, where this notation is used for the spectral norm and the entrywise 2-norm is the Frobenius norm.) Finally, the standard inner product between vectors x, y is $\langle x, y\rangle = x^T y$; e_i are standard basis vectors of the relevant dimension; I_n denotes the $n \times n$ identity matrix; and c refers to a generic constant whose specific value may vary throughout the paper.

Two Useful Tail Inequalities. The following two Bernstein-type tail inequalities are useful because they give tail bounds without reference to the number of i.i.d. trials. The first bound is due to Maurer [12], and the second is an immediate application of the first.

Lemma 2.1. ([12]) *Let $X_i \geq 0$ be independent random variables with $\sum_i \mathbf{E}[X_i^2] < \infty$, and define $X = \sum_i X_i$. Then, for any $t > 0$,*

$$\mathbf{Pr}[X \leq \mathbf{E}[X] - t] \leq \exp\left(\frac{-t^2}{2\sum_i \mathbf{E}[X_i^2]}\right).$$

Lemma 2.2. *Let x_i be i.i.d. Bernoulli random variables with probability p, and let $X = \sum_{i\in[n]} \xi_i x_i$, where $\xi_i \geq 0$, with $\sum_{i\in[n]} \xi_i = \xi$ and $\sum_{i\in[n]} \xi_i^2 \leq \xi^2/\beta^2$. Then, for any $t > 0$,*

$$\mathbf{Pr}[X \geq \xi(p + t)] \leq \exp\left(-\frac{\beta^2 t^2}{2(1-p)}\right).$$

Sums of Cauchy Random Variables. The Cauchy distribution, having density $p(x) = \frac{1}{\pi}\frac{1}{1+x^2}$, is the unique 1-stable distribution. If $C_1, \ldots, C_M$ are independent Cauchys, then $\sum_{i\in[M]} \gamma_i C_i$ is distributed as a Cauchy scaled by $\gamma = \sum_{i\in[M]} |\gamma_i|$. The Cauchy distribution will factor heavily in our discussion, and bounds for sums of Cauchy random variables will be used throughout.

Lemma 2.3. *For $i \in [m]$, let C_i be m (not necessarily independent) Cauchy random variables, and $\gamma_i > 0$ with*

$\gamma = \sum_{i\in[m]} \gamma_i$. *Let* $X = \sum_{i\in[m]} \gamma_i |C_i|$. *Then, for any* $t \geq \gamma$,

$$\begin{aligned}\mathbf{Pr}\left[X > \gamma t\right] &\leq \frac{2}{\pi t}\left(\frac{\log(1+(mt)^2)}{1-2/\pi t}+1\right)\\ &= \frac{4\log(mt)}{\pi t}\left(1+o(1)\right).\end{aligned}$$

Remark. The bound has only logarithmic dependence on the number of Cauchy random variables and does *not* rely on any independence assumption among the random variables. Even if the Cauchys are independent, one cannot substantially improve on this bound due to the nature of the Cauchy distribution. This is because, for independent Cauchys, $\sum_i \gamma_i |C_i| \geq |\sum_i \gamma_i C_i|$, and the latter sum is itself distributed as a Cauchy scaled by γ. Hence for independent Cauchys, $\mathbf{Pr}[X \geq \gamma t] \geq \frac{2}{\pi}\tan^{-1} t = \Omega(\frac{1}{t})$.

LEMMA 2.4. *For* $i \in [r]$, *let* C_i *be independent Cauchy random variables, and* $\gamma_i \geq 0$ *with* $\gamma = \sum_{i\in[r]} \gamma_i$ *and* $\sum_{i\in[r]} \gamma_i^2 \leq \gamma^2/\beta^2$. *Let* $X = \sum_{i\in[r]} \gamma_i |C_i|$. *Then, for any* $t \geq 0$,

$$\mathbf{Pr}\left[X \leq \gamma(1-t)\right] \leq \exp\left(-\frac{\beta^2 t^2}{3}\right).$$

An ℓ_1 Sampling Lemma. We will also need an "ℓ_1-sampling lemma," which is an application of Bernstein's inequality. This lemma bounds how ℓ_1 norms get distorted under sampling according to ℓ_1 probabilities.

LEMMA 2.5. *Let* $Z \in \mathbb{R}^{n\times k}$ *and suppose that for* $i \in [n]$, $a\|Z_{(i)}\|_1 \leq \lambda_i \leq b\|Z_{(i)}\|_1$. *For* $s > 0$, *define* $\hat{p}_i = \min\{1, s\cdot\lambda_i/\sum_{i\in[n]}\lambda_i\}$, *and let* $D \in \mathbb{R}^{n\times n}$ *be a random diagonal matrix with* $D_{ii} = 1/\hat{p}_i$ *with probability* $\hat{p}_i$, *and* 0 *otherwise. Then, for any (fixed)* $x \in \mathbb{R}^k$, *with probability at least* $1-\delta$,

$$(1-\varepsilon)\|Zx\|_1 \leq \|DZx\|_1 \leq (1+\varepsilon)\|Zx\|_1,$$

where $\delta \leq 2\exp\left(\frac{-\frac{a}{b}s\varepsilon^2\|Zx\|_1}{(2+\frac{2}{3}\varepsilon)\|Z\|_1\|x\|_\infty}\right)$.

3 Main Technical Result: the Fast Cauchy Transform

In this section, we present the Fast Cauchy Transform (FCT), which is an ℓ_1-based analog of the fast Johnson-Lindenstrauss transform (FJLT). We will actually present two related constructions, one based on using a quickly-constructable low-coherence matrix, and one based on using a version of the FJLT. In both cases, these matrices will be rescaled by Cauchy random variables (hence the name *Fast Cauchy Transform*). We will also state our main results, Theorems 3.1 and 3.2, which provides running time and quality-of-approximation guarantees for these two FCT embeddings.

3.1 FCT1 Construction: via a Low-coherence Matrix This FCT construction first preprocesses by a deterministic low-coherence "spreading matrix," then rescales by Cauchy random variables, and finally samples linear combinations of the rows. Let $\delta \in (0,1]$ be a parameter governing the failure probability of our algorithm. Then, we construct Π_1 as

$$\Pi_1 \equiv 4BC\tilde{H},$$

where:

B $\in \mathbb{R}^{r_1\times 2n}$ has each column chosen independently and uniformly from the r_1 standard basis vectors for $\mathbb{R}^{r_1}$; for α sufficiently large, we will set the parameter $r_1 = \alpha d \log\frac{d}{\delta}$, where δ controls the probability that our algorithms fail and α is a suitably large constant;

$C \in \mathbb{R}^{2n\times 2n}$ is a diagonal matrix with diagonal entries chosen independently from a Cauchy distribution; and

$\tilde{H}$ $\in \mathbb{R}^{2n\times n}$ is a block-diagonal matrix comprised of n/s blocks along the diagonal. Each block is the $2s\times s$ matrix $G_s \equiv \left[\begin{smallmatrix}H_s\\ I_s\end{smallmatrix}\right]$, where I_s is the $s\times s$ identity matrix, and H_s is the normalized Hadamard matrix. We will set $s = r_1^6$. (Here, for simplicity, we assume s is a power of two and n/s is an integer.)

$$\tilde{H} \equiv \begin{bmatrix} G_s & & & \\ & G_s & & \\ & & \ddots & \\ & & & G_s \end{bmatrix}$$

(For completeness, we remind the reader that the (non-normalized) $n\times n$ matrix of the Hadamard transform H_n may be defined recursively as follows:

$$H_n = \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix}, \quad \text{with} \quad H_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}.$$

The $n\times n$ normalized matrix of the Hadamard transform is then equal to $\frac{1}{\sqrt{n}}H_n$; hereafter, we will denote this normalized matrix by H_n.) Heuristically, the effect of $\tilde{H}$ in the above FCT construction is to spread the weight of a vector, so that $\tilde{H}y$ has many entries that are not too small. This means that the vector $C\tilde{H}y$ comprises Cauchy random variables with scale factors that are not too small; and finally these variables are summed up by

B, yielding a vector $BC\tilde{H}y$, whose ℓ_1 norm won't be too small relative to $\|y\|_1$. For this version of the FCT, we have the following theorem.

THEOREM 3.1. (FAST CAUCHY TRANSFORM (FCT1)) *There is a distribution (given by the above construction) over matrices $\Pi_1 \in \mathbb{R}^{r_1 \times n}$, with $r_1 = O(d \log d + d \log \frac{1}{\delta})$, such that for an arbitrary (but fixed) $A \in \mathbb{R}^{n \times d}$, and for all $x \in \mathbb{R}^d$, the inequalities*

$$\|Ax\|_1 \leq \|\Pi_1 Ax\|_1 \leq \kappa \|Ax\|_1 \qquad (3.2)$$

hold with probability $1 - \delta$, where

$$\kappa = O\left(\frac{d\sqrt{s}}{\delta} \log(r_1 d)\right).$$

Further, for any $y \in \mathbb{R}^n$, the product $\Pi_1 y$ can be computed in $O(n \log r_1)$ time.

Setting δ to a small constant, since $\sqrt{s} = r_1^3$ and $r_1 = O(d \log d)$, it follows that $\kappa = O(d^4 \log^4 d)$ in the above theorem.

Remark. The existence of such a Π_1 satisfying bounds of the form (3.2) was established by Sohler and Woodruff [13]. Here, our contribution is to show that Π_1 can be factored into structured matrices so that the product $\Pi_1 A$ can be computed in $O(nd \log d)$ time. We also remark that, in additional theoretical bounds provided by the FJLT, high-quality numerical implementations of variants of the Hadamard transform exist, which is an additional plus for our empirical evaluations of Theorem 3.1 and Theorem 3.2.

Remark. Our proof of the upper bound uses Lemma 2.3; and our proof of the lower bound uses a tail bound for $\|Bg\|_2^2$ in terms of $\|g\|_2$ and $\|g\|_1$, where g is any positive vector in $\mathbb{R}^n$, and B is the matrix used in our FCT construction. $\|Bg\|_2^2 = \sum_j \gamma_j^2$ where $\gamma_j = \sum_i B_{ji} g_i$ are *anti-correlated* random variables. To get concentration, we independently bounded γ_j^2 using Lemma 2.2. Finally we use Lemma 2.4, which leads to $s = r^6$ to obtain the high probability result; this resulted in the bound $\kappa = O(d^4 \log^4 d)$.

3.2 FCT2 Construction: via a Fast Johnson-Lindenstrauss Transform This FCT construction first preprocesses by a FJLT and then rescales by Cauchy random variables. Recall that $\delta \in (0, 1]$ is a parameter governing the failure probability of our algorithm; and let $\eta > 0$ be a generic arbitrarily small positive constant (whose value may change from one formula to another). Let $r_1 = c \cdot d \log \frac{d}{\delta}$, $s = c' \cdot (d + \log \frac{n}{\delta})$, and $t = s^{2+\eta}$, where the parameters $c, c' > 0$ are appropriately large constants. Then, we construct $\Pi_1 \in \mathbb{R}^{r_1 \times n}$ as

$$\Pi_1 \equiv \frac{8}{r_1}\sqrt{\frac{\pi t}{2s}} \cdot C\tilde{H},$$

where:

C $\in$ $\mathbb{R}^{r_1 \times ns/t}$ is a matrix of independent Cauchy random variables; and

$\tilde{H}$ $\in$ $\mathbb{R}^{ns/t \times n}$ is a block-diagonal matrix comprising n/t blocks along the diagonal. Each block is the $s \times t$ Fast Johnson-Lindenstrauss matrix G. Here, for simplicity, we assume that n/t is an integer.

$$\tilde{H} \equiv \begin{bmatrix} G & & & \\ & G & & \\ & & \ddots & \\ & & & G \end{bmatrix}.$$

Informally, the matrix $\tilde{H}$ reduces the dimensionality of the input space by a very small amount such that the "slow" Cauchy Transform C of [13] can be applied in the allotted time. Then, since we are ultimately multiplying by C, the results of [13] still hold; but since the dimensionality is slightly reduced, the running time is improved. For this version of the FCT, we have the following theorem.

THEOREM 3.2. (FAST CAUCHY TRANSFORM (FCT2)) *There is a distribution (given by the above construction) over matrices $\Pi_1 \in \mathbb{R}^{r_1 \times n}$, with $r_1 = O(d \log \frac{d}{\delta})$, such that for arbitrary (but fixed) $A \in \mathbb{R}^{n \times d}$, and for all $x \in \mathbb{R}^d$, the inequalities*

$$\|Ax\|_1 \leq \|\Pi_1 Ax\|_1 \leq \kappa \|Ax\|_1$$

hold with probability $1 - \delta$, where $\kappa = O(\frac{d}{\delta}(d + \log \frac{n}{\delta})^{1+\eta} \log d)$. Further, for any $y \in \mathbb{R}^n$, the product $\Pi_1 y$ can be computed in $O(n \log \frac{d}{\delta})$ time.

Setting δ to be a small constant and for $\log n < d$, $r_1 = O(d \log d)$, $\kappa = O(d^{2+\eta} \log d)$ and $\Pi_1 A$ can be computed in $O(nd \log d)$ time. Thus, we have a fast linear oblivious mapping from from $\ell_1^n \mapsto \ell_1^{O(d \log d)}$ that has distortion $O(d^{2+\eta} \log d)$ on any (fixed) d-dimensional subspace of $\mathbb{R}^n$.

Remark. For $\log n < d$, FCT2 gives a better dependence of the distortion on d, but more generally FCT2 has a dependence on $\log n$. This dependence arises because the random FJLT matrix does not give a deterministic guarantee for spreading out a vector whereas the low coherence matrix used in FCT1 does give a deterministic guarantee. This means that in using the union bound, we need to overcome a factor of n.

Remark. The requirement $t \geq s^{2+\eta}$ is set by a restriction in a technical lemma in the proof of Theorem 3.2. If a stronger version of this lemma can be proved that relaxes the restriction $t \geq s^{2+\eta}$, then correspondingly the bound of Theorem 3.2 will improve.

Remark. This second construction has the benefit of being easily extended to constructing well-conditioned bases of ℓ_p, for $p > 1$; see Section 5.

4 Algorithmic Applications in ℓ_1 of the FCT

In this section, we describe three related applications of the FCT to ℓ_1-based problems. The first is to the fast construction of an ℓ_1 well-conditioned basis and the fast approximation of ℓ_1 leverage scores; the second is a fast algorithm for the least absolute deviations or ℓ_1 regression problem; and the third is to a fast algorithm for the ℓ_1 norm subspace approximation problem.

4.1 Fast Construction of an ℓ_1 Well-conditioned Basis and ℓ_1 Leverage Scores We start with the following definition, adapted from [5], of a basis that is "good" for the ℓ_1 norm in a manner that is analogous to how an orthogonal matrix is "good" for the ℓ_2 norm.

Definition 2. *A basis U for the range of A is (α, β)-conditioned if $\|U\|_1 \leq \alpha$ and for all $x \in \mathbb{R}^d$, $\|x\|_\infty \leq \beta \|Ux\|_1$. We will say that U is* well-conditioned *if α and β are low-degree polynomials in d, independent of n.*

Remark. An Auerbach basis for A is $(d, 1)$-conditioned, and thus we know that there exist well-conditioned bases for ℓ_1. More generally, well-conditioned bases can be defined in any ℓ_p norm, using the notion of a dual norm ℓ_p^*, and these have proven important for solving ℓ_p regression problems [5]. Our focus in this section is the ℓ_1 norm, for which the dual norm is the ℓ_∞ norm, but in Section 5 we will return to a discussion of extensions to the ℓ_p norm.

Our main algorithm for constructing an ℓ_1 well-conditioned basis, FastL1Basis, is summarized in Figure 1. This algorithm was originally presented in [13], and our main contribution here is to improve its running time. Given an $n \times d$ matrix A, let $\Pi_1 \in \mathbb{R}^{r_1 \times n}$ be any projection matrix such that for any $x \in \mathbb{R}^d$,

$$\|Ax\|_1 \leq \|\Pi_1 Ax\|_1 \leq \kappa \|Ax\|_1. \tag{4.3}$$

For example, it could be constructed with either of the FCT constructions described in Section 3, or with the "slow" Cauchy Transform of [13], or via some other means. After computing the matrix Π_1, the FastL1Basis algorithm of Figure 1 consists of the following steps: construct $\Pi_1 A$ and an R such that $\Pi_1 A = QR$, where Q has orthonormal columns (for example using a QR-factorization of $\Pi_1 A$); and then return $U = AR^{-1} = A(Q^T \Pi_1 A)^{-1}$.

```
FastL1Basis($A$):
 1: Let $\Pi_1$ be an $r_1 \times n$ matrix satisfying (4.3),
    e.g., as constructed with one of the FCTs of
    Section 3.
 2: Compute $\Pi_1 A \in \mathbb{R}^{r_1 \times d}$ and its QR-
    factorization: $\Pi_1 A = QR$, where $Q$ is an or-
    thogonal matrix, i.e., $Q^T Q = I$.
 3: Return $U = AR^{-1} = A(Q^T \Pi_1 A)^{-1}$
```

Figure 1: Our main algorithm for the fast construction of an ℓ_1 well-conditioned basis of an $n \times d$ matrix A. Note the structural similarities with the algorithm of [6] for computing quickly approximations to the ℓ_2 leverage scores and an ℓ_2 well-conditioned basis.

The next theorem and its corollary are our main results for the FastL1Basis algorithm; and this theorem follows by combining our Theorem 3.2 with Theorems 9 and 10 of [13].

Theorem 4.1. *For any $A \in \mathbb{R}^{n \times d}$, the basis $U = AR^{-1}$ constructed by FastL1Basis(A) of Figure 1 using any Π_1 satisfying (4.3) is a $(d\sqrt{r_1}, \kappa)$-conditioned basis for the range of A.*

Corollary 4.1. *If Π_1 is obtained from the FCT2 construction of Theorem 3.2, then the resulting U is an (α, β)-conditioned basis for A, with $\alpha = O(d^{3/2} \log^{1/2} d)$ and $\beta = O(d^{2+\eta} \log d)$, with probability $1 - \delta$. The time to compute the change of basis matrix R^{-1} is $O(nd \log d + d^3 \log d)$.*

Remark. Our constructions that result in Π_1 satisfying (4.3) do not require that $A \in \mathbb{R}^{n \times d}$; they only require that A have rank d, and so can be applied to any $A \in \mathbb{R}^{n \times m}$ having rank d. In this case, a small modification is needed in the construction of U, because $R \in \mathbb{R}^{d \times m}$, and so we need to use $R^\dagger$ instead of R^{-1}. The running time will involve terms with m. This can be improved by processing A quickly into a smaller matrix by sampling columns so that the range is preserved (as in [13]), which we do not discuss further.

The notion of a well-conditioned basis plays an important role in our subsequent algorithms. Basically, the reason is that these algorithms compute approximate answers to the problems of interest (either the ℓ_1 regression problem or the ℓ_1 subspace approximation problem) by using information in that basis to construct a nonuniform importance sampling distribution with which to randomly sample. This motivates the following definition.

Definition 3. *Given a well-conditioned basis U for the range of A, let the n-dimensional vector $\tilde{\lambda}$, with elements defined as $\tilde{\lambda}_i = ||U_{(i)}||_1$, be the ℓ_1* leverage scores *of A.*

Remark. The name ℓ_1 *leverage score* is by analogy with the ℓ_2 *leverage scores*, which are important in random sampling algorithms for ℓ_2 regression and low-rank matrix approximation [11, 10, 6]. As with ℓ_2 regression and low-rank matrix approximation, our result for ℓ_1 regression and ℓ_1 subspace approximation will ultimately follow from the ability to approximate these scores quickly. Note, though, that these ℓ_1-based scores are not well-defined for a given matrix A, in the sense that the ℓ_1 norm is not rotationally invariant, and thus depending on the basis that is chosen, these scores can differ by factors that depend on low-degree polynomials in d. This contrasts with ℓ_2, since for ℓ_2 any orthogonal matrix spanning a given subspace leads to the same ℓ_2 leverage scores. We will tolerate this ambiguity since these ℓ_1 leverage scores will be used to construct an importance sampling distribution, and thus up to low-degree polynomial factors in d, which our analysis will take into account, it will not matter.

4.2 Main Algorithm for Fast ℓ_1 Regression

Here, we consider the ℓ_1 regression problem, also known as the *least absolute deviations* problem, the goal of which is to minimize the ℓ_1 norm of the residual vector $Ax - b$. That is, given as input a design matrix $A \in \mathbb{R}^{n\times d}$, with $n > d$, and a response or target vector $b \in \mathbb{R}^n$, compute

$$\mathcal{Z} = \min_{x\in\mathbb{R}^d} ||b - Ax||_1, \tag{4.4}$$

and an x^* achieving this minimum.

Prior work has shown that there is a diagonal sampling matrix D with a small number of nonzero entries so that $\hat{x} = \mathrm{argmin}_{x\in\mathbb{R}^d} \|D(Ax-b)\|_1$ satisfies

$$\|A\hat{x} - b\|_1 \le (1+\varepsilon)\|Ax^* - b\|_1,$$

where x^* is the optimal solution for the minimization in (4.4); see [5, 13]. The matrix D can be found by sampling its diagonal entries independently according to a set of probabilities p_i that are proportional to the ℓ_1 leverage scores. Here, we give a fast algorithm to compute estimates $\hat{p}_i$ of these probabilities. This permits us to develop an improved algorithm for ℓ_1 regression and to construct efficiently a small coreset for an arbitrary ℓ_1 regression problem.

In more detail, Figure 2 presents the FastCauchyRegression algorithm, which we summarize here. Let $X = \begin{bmatrix} A & -b \end{bmatrix}$. First, a matrix Π_1 satisfying (4.3) is used to reduce the dimensionality of X to $\Pi_1 X$ and to obtain the orthogonalizer R^{-1}. Let $U = XR^{-1}$ be the resulting well-conditioned basis for the range of X. The probabilities we use to sample rows are essentially the row-norms of U. However, to compute XR^{-1} explicitly takes $O(nd^2)$ time, which is already too costly, and so we need to estimate $\|U_{(i)}\|_1$ without explicitly computing U. To construct these probabilities quickly, we use a second random projection Π_2—on the *right*. This second projection allows us to estimate the norms of the rows of XR^{-1} efficiently to within relative error (which is all we need) using the median of r_2 independent Cauchy's, each scaled by $||U_{(i)}||_1$. (Note that this is similar to what was done in [6] to approximate the ℓ_2 leverage scores of an input matrix.) These probabilities are then used to construct a carefully down-sampled (and rescaled) problem, the solution to which will give us our $(1+\varepsilon)$ approximation to the original problem.

The next theorem summarizes our main quality-of-approximation results for the FastCauchyRegression algorithm of Figure 2. It improves the $O(nd^2 + \mathrm{poly}(d\varepsilon^{-1}\log n))$ algorithm of [13], which in turn improved the result in [5]. (Technically, the running time of [13] is $O(nd^{\omega^+-1} + \mathrm{poly}(d\varepsilon^{-1}\log n))$, where ω^+ is any constant larger than the exponent for matrix multiplication; for practical purposes, we can set $\omega^+ = 3$.) Our improved running time comes from using the FCT and a simple row-norm estimator for the row-norms of a well-conditioned basis.

Theorem 4.2. *Given are $\varepsilon \in (0,1)$, $\rho > 0$, $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$. FastCauchyRegression(A,b) constructs a coreset specified by the diagonal sampling matrix D and a solution vector $\hat{x} \in \mathbb{R}^d$ that minimizes the weighted regression objective $\|D(Ax-b)\|_1$. The solution $\hat{x}$ satisfies, with probability at least $1 - \frac{1}{d^\rho}$,*

$$\|A\hat{x} - b\|_1 \le \left(\frac{1+\varepsilon}{1-\varepsilon}\right)\|Ax - b\|_1, \qquad \forall x \in \mathbb{R}^d.$$

Further, with probability $1 - o(1)$, the entire algorithm to construct $\hat{x}$ runs in time

$$O\left(nd\log n + \phi(s,d)\right) = O\left(nd\log n + \tfrac{1}{\varepsilon^2}\,\mathrm{poly}(d, \log \tfrac{d}{\varepsilon})\right),$$

where $\phi(s,d)$ is the time to solve an ℓ_1-regression problem on s vectors in d dimensions, and if FCT2 is used to construct Π_1 then $s = O\left(\frac{1}{\varepsilon^2} d^{\rho+\frac{9}{2}+\eta}\log^{\frac{3}{2}}(\frac{d}{\varepsilon})\right)$.

Remarks. Several remarks about our results for the ℓ_1 regression problem are in order.

- Our proof analyzes a more general problem $\min_{x\in\mathcal{C}} ||Xx||_1$. In order to get the result, we need to preserve norms under sampling, which is what

```
FastCauchyRegression$(A, b)$:
1: Let $X = \begin{bmatrix} A & -b \end{bmatrix} \in \mathbb{R}^{n\times(d+k)}$ and construct $\Pi_1$, an $r_1 \times n$ matrix satisfying (4.3) with $A$ replaced
   by $X$. (If $b$ is a vector then $k = 1$.)
2: Compute $X' = \Pi_1 X \in \mathbb{R}^{r_1\times(d+k)}$ and its QR factorization, $\Pi_1 X = QR$. (Note that $\Pi_1 X R^{-1}$ has
   orthonormal columns.)
3: Let $\Pi_2 \in \mathbb{R}^{(d+k)\times r_2}$ be a matrix of independent Cauchys, with $r_2 = 15 \log \frac{2n}{\delta}$.
4: Let $U = XR^{-1}$ and construct $\Lambda = U\Pi_2 \in \mathbb{R}^{n\times r_2}$.
5: For $i \in [n]$, compute $\lambda_i = \mathrm{median}_{j\in r_2} |\Lambda_{ij}|$.
6: For $i \in [n]$ and $s = \frac{63\kappa(d+k)\sqrt{r_1}}{\varepsilon^2}\left((d+k)\log\frac{4d\sqrt{r_1}\max((d+k)\sqrt{r_1},\kappa)}{\varepsilon} + \log\frac{2}{\delta}\right)$, compute probabilities
   $\hat{p}_i = \min\left\{1, s \cdot \frac{\lambda_i}{\sum_{i\in[n]}\lambda_i}\right\}$.
7: Let $D \in \mathbb{R}^{n\times n}$ be diagonal with independent entries: $D_{ii} = \begin{cases} \frac{1}{\hat{p}_i} & \text{prob. } \hat{p}_i; \\ 0 & \text{prob. } 1 - \hat{p}_i. \end{cases}$
8: Return $\hat{x} \in \mathbb{R}^d$ that minimizes $\|DAx - Db\|_1$ w.r.t. $x$ (using linear programming).
```

Figure 2: Algorithm for solving ℓ_1 regression.

Lemma 2.5 allows us to do. We mention that our methods extend with minor changes to ℓ_p regression, for $p > 1$. This is discussed in Section 5.

- A natural extension of our algorithm to matrix-valued right hand sides b gives a $(1 + \varepsilon)$ approximation in a similar running time for the ℓ_1-norm subspace approximation problem. See Section 4.3 for details.

- We can further improve the efficiency of solving this simple ℓ_1 regression problem, thereby replacing the $nd \log n$ running time term in Theorem 4.2 with $nd \log(d\varepsilon^{-1} \log n)$, but at the expense of a slightly larger sample size s. The improved algorithm is essentially the same as the FastCauchyRegression algorithm, except with two differences: Π_2 is chosen to be a matrices of i.i.d. Gaussians, for a value $r_2 = O(\log(d\varepsilon^{-1} \log n))$; and, to accommodate this, the size of s needs to be increased. Details can be found in the technical report version of this conference paper [4].

4.3 ℓ_1 norm Subspace Approximation Finally, we consider the ℓ_1 *norm subspace approximation problem*: Given the n points in the $n \times d$ matrix A and a parameter $k \in [d-1]$, embed these points into a subspace of dimension k to obtain the embedded points $\hat{A}$ such that $\|A - \hat{A}\|_1$ is minimized. (Note that this is the ℓ_1 analog of the ℓ_2 problem that is solved by the Singular Value Decomposition.) When $k = d - 1$, the subspace is a hyperplane, and the task is to find the hyperplane passing through the origin so as to minimize the sum of ℓ_1 distances of the points to the hyperplane. In order to solve this problem with the methods from Section 4.2, we take advantage of the observation made in [2] (see also Lemma 18 of [13]) that this problem can be reduced to d related ℓ_1 regressions of A onto each of its columns, a problem sometimes called *multiple regression*. We can extend our ℓ_1 "simple" regression algorithm to an ℓ_1 "multiple" regression algorithm; and this can be used to solve the ℓ_1 norm subspace approximation problem. Details can be found in the technical report version of this conference paper [4].

5 Extensions to ℓ_p, for $p > 1$

In this section, we describe extensions of our methods to ℓ_p, for $p > 1$. We will first (in Section 5.1) discuss ℓ_p norm conditioning and connect it to ellipsoidal rounding, followed by a fast rounding algorithm for general centrally symmetric convex sets (in Section 5.2); and we will then (in Section 5.3) show how to obtain quickly a well-conditioned basis for the ℓ_p norm, for any $p \in [1, \infty)$ and (in Section 5.4) show how this basis can be used for improved ℓ_p regression. These results will generalize our results for ℓ_1 from Sections 4.1 and 4.2, respectively, to general ℓ_p.

5.1 ℓ_p norm Conditioning and Ellipsoidal Rounding As with ℓ_2 regression, ℓ_p regression problems are easier to solve when they are well-conditioned. Thus, we start with the definition of the ℓ_p norm condition number κ_p of a matrix A.

Definition 4. *Given an $n \times d$ matrix A, let*

$$\sigma_p^{\max}(A) = \max_{\|x\|_2 \le 1} \|Ax\|_p \text{ and } \sigma_p^{\min}(A) = \min_{\|x\|_2 \ge 1} \|Ax\|_p.$$

Then, we denote by $\kappa_p(A)$ the ℓ_p norm condition number *of A, defined to be:*

$$\kappa_p(A) = \sigma_p^{\max}(A)/\sigma_p^{\min}(A).$$

For simplicity, we will use κ_p, $\sigma_p^{\min}$, and $\sigma_p^{\max}$ when the underlying matrix is clear.

There is a strong connection between the ℓ_p norm condition number and the concept of an (α, β, p)-conditioning developed by Dasgupta et al. [5].

Definition 5. ([5])) *Given an $n \times d$ matrix A and $p \in [1, \infty]$, let q be the dual norm of p. Then A is (α, β, p)*-conditioned *if (1) $\|A\|_p \le \alpha$, and (2) for all $z \in \mathbb{R}^d$, $\|z\|_q \le \beta \|Az\|_p$. Define $\bar{\kappa}_p(A)$ as the minimum value of $\alpha\beta$ such that A is (α, β, p)-conditioned. We say that A is* p-well-conditioned *if $\bar{\kappa}_p(A) = \mathcal{O}(\mathrm{poly}(d))$, independent of n.*

The following lemma characterizes the relationship between these two quantities.

Lemma 5.1. *Given an $n \times d$ matrix A and $p \in [1, \infty]$, we always have*

$$d^{-|1/2-1/p|}\kappa_p(A) \le \bar{\kappa}_p(A) \le d^{\max\{1/2, 1/p\}}\kappa_p(A).$$

Although it is easier to describe sampling algorithms in terms of $\bar{\kappa}_p$, after we show the equivalence between κ_p and $\bar{\kappa}_p$, it will be easier for us to discuss conditioning algorithms in terms of κ_p, which naturally connects to ellipsoidal rounding algorithms.

Definition 6. *Let $\mathcal{C} \subseteq \mathbb{R}^d$ be a convex set that is full-dimensional, closed, bounded, and centrally symmetric with respect to the origin. An ellipsoid $\mathcal{E} = \{x \in \mathbb{R}^d \,|\, \|Rx\|_2 \le 1\}$ is a* κ-rounding *of $\mathcal{C}$ if it satisfies $\mathcal{E}/\kappa \subseteq \mathcal{C} \subseteq \mathcal{E}$, for some $\kappa \ge 1$, where $\mathcal{E}/\kappa$ means shrinking $\mathcal{E}$ by a factor of $1/\kappa$.*

To see the connection between rounding and conditioning, let $\mathcal{C} = \{x \in \mathbb{R}^d \,|\, \|Ax\|_p \le 1\}$ and assume that we have a κ-rounding of $\mathcal{C}$: $\mathcal{E} = \{x \,|\, \|Rx\|_2 \le 1\}$. This implies

$$\|Rx\|_2 \le \|Ax\|_p \le \kappa\|Rx\|_2, \quad \forall x \in \mathbb{R}^d.$$

If we let $y = Rx$, then we get

$$\|y\|_2 \le \|AR^{-1}y\|_p \le \kappa\|y\|_2, \quad \forall y \in \mathbb{R}^d.$$

Therefore, we have $\kappa_p(AR^{-1}) \le \kappa$. So a κ-rounding of $\mathcal{C}$ leads to a κ-conditioning of A.

5.2 Fast Ellipsoidal Rounding Here, we provide a deterministic algorithm to compute a $2d$-rounding of a centrally symmetric convex set in $\mathbb{R}^d$ that is described by a separation oracle. Recall the well-known result due to John [8] that for a centrally symmetric convex set $\mathcal{C}$ there exists a $d^{1/2}$-rounding and that such rounding is given by the Löwner-John (LJ) ellipsoid of $\mathcal{C}$, *i.e.*, the minimal-volume ellipsoid containing $\mathcal{C}$. However, finding this $d^{1/2}$-rounding is a hard problem. To state algorithmic results, suppose that $\mathcal{C}$ is described by a separation oracle and that we are provided an ellipsoid $\mathcal{E}_0$ that gives an L-rounding for some $L \ge 1$. In this case, the best known algorithmic result of which we are aware is that we can find a $(d(d+1))^{1/2}$-rounding in polynomial time, in particular, in $O(d^4 \log L)$ calls to the oracle; see Lovász [9, Theorem 2.4.1]. This result was used by Clarkson [3] and by Dasgupta *et al.* [5]. Here, we follow the same construction, but we show that it is much faster to find a (slightly worse) $2d$-rounding.

Theorem 5.1. *Given a centrally symmetric convex set $\mathcal{C} \subseteq \mathbb{R}^d$ centered at the origin and described by a separation oracle, and an ellipsoid $\mathcal{E}_0$ centered at the origin such that $\mathcal{E}_0/L \subseteq \mathcal{C} \subseteq \mathcal{E}_0$ for some $L \ge 1$, it takes at most $3.15d^2 \log L$ calls to the oracle and additional $O(d^4 \log L)$ time to find a $2d$-rounding of $\mathcal{C}$.*

Applying Theorem 5.1 to the convex set $\mathcal{C} = \{x \,|\, \|Ax\|_p \le 1\}$, with the separation oracle described via a subgradient of $\|Ax\|_p$ and the initial rounding provided by the "R" matrix from the QR decomposition of A, we improve the running time of the algorithm used by Clarkson [3] and by Dasgupta *et al.* [5] from $\mathcal{O}(nd^5 \log n)$ to $\mathcal{O}(nd^3 \log n)$ while maintaining an $\mathcal{O}(d)$-conditioning.

Theorem 5.2. *Given an $n \times d$ matrix A with full column rank, it takes at most $\mathcal{O}(nd^3 \log n)$ time to find a matrix $R \in \mathbb{R}^{d \times d}$ such that $\kappa_p(AR^{-1}) \le 2d$.*

5.3 Fast Construction of an ℓ_p Well-conditioned Basis Here, we consider the construction of a basis that is well-conditioned for ℓ_p. To obtain results for general ℓ_p that are analogous to those we obtained for ℓ_1, we will extend the FCT2 construction from Section 3.2, combined with Theorem 5.1.

Our main algorithm for constructing a p-well-conditioned basis, the FastLpBasis algorithm, is summarized in Figure 3. The algorithm first applies block-wise embeddings in the ℓ_2 norm, similar to the construction of FCT2; it then uses the algorithm of Theorem 5.1 to compute a $(2d)$-rounding of a special convex set $\tilde{\mathcal{C}}$ and obtain the matrix R. It is thus a generalization of our FastL1Basis algorithm of Section 4.1, and it follows the

```
FastLpBasis($A$):
1: Let $s = \Theta(d + \log n)$, $t = \Theta(sd^2)$, and $G$ be an $s \times t$ Fast Johnson-Lindenstrauss matrix, the same as the matrix $G$ in the FCT2 construction.
2: Partition $A$ along its rows into sub-matrices of size $t \times d$, denoted by $A_1, \ldots, A_N$, compute $\tilde{A}_i = GA_i$ for $i = 1, \ldots, N$, and define
   $\tilde{\mathcal{C}}=\left\{ x \,\middle|\, \left( \sum_{i=1}^{N} \|\tilde{A}_i x\|_2^p \right)^{1/p} \le 1 \right\}$, and $\tilde{A}=\begin{pmatrix} \tilde{A}_1 \\ \vdots \\ \tilde{A}_N \end{pmatrix}$.
3: Apply the algorithm of Theorem 5.1 to obtain a $(2d)$-rounding of $\tilde{\mathcal{C}}$: $\mathcal{E} = \{x \mid \|Rx\|_2 \le 1\}$.
4: Output $AR^{-1}$.
```

Figure 3: Our main algorithm for the fast construction of an ℓ_p well-conditioned basis of an $n \times d$ matrix A. Note the structural similarities with our FastL1Basis algorithm of Figure 1 for computing quickly an ℓ_1 well-conditioned basis.

same high-level structure laid out by the algorithm of [6] for computing approximations to the ℓ_2 leverage scores and an ℓ_2 well-conditioned basis.

The next theorem is our main result for the FastLpBasis algorithm. It improves the running time of the algorithm of Theorem 5.2, at the cost of slightly worse conditioning quality. However, these worse factors will only contribute to a low-order additive poly(d) term in the running time of our ℓ_p regression application in Section 5.4.

THEOREM 5.3. *For any $A \in \mathbb{R}^{n\times d}$ with full column rank, the basis AR^{-1} constructed by FastLpBasis(A) (Figure 3), with probability at least $1 - 1/n$, is ℓ_p well-conditioned with $\kappa_p(AR^{-1}) = \mathcal{O}(dt^{|1/p-1/2|})$. The time to compute R is $O(nd \log n)$.*

When $d > \log n$, $\kappa_p(AR^{-1}) = \mathcal{O}(d^{1+3\cdot|1/p-1/2|})$ and hence $\bar{\kappa}_p(AR^{-1}) = \mathcal{O}(d^{1+3\cdot|1/p-1/2|+\max\{1/p,1/2\}})$ by Lemma 5.1. Note that, even for the case when $p = 1$, we have $\bar{\kappa}_p(AR^{-1}) = \mathcal{O}(d^{7/2})$, which is slightly better than FCT2 (see Corollary 4.1). However, we have to solve a rounding problem of size $ns/t \times d$ in the step 2 of FastLpBasis, which requires storage and work depending on n.

5.4 Fast ℓ_p Regression Here, we show that the overconstrained ℓ_p regression problem can be solved with a generalization of the algorithms of Section 4.2 for solving ℓ_1 regression; we will call this generalization the FastLpRegression algorithm. In particular, as with the algorithm for ℓ_1 regression, this FastLpRegression algorithm for the ℓ_p regression problem uses an ℓ_p well-conditioned basis and samples rows of A with probabilities proportional to the ℓ_p norms of the rows of the corresponding well-conditioned basis (which are the ℓ_p analogs of the ℓ_2 leverage scores). As with the FastCauchyRegression, this entails using—for speed—a second random projection Π_2 applied to AR^{-1}—on the right—to estimate the row norms. This allows fast estimation of the ℓ_2 norms of the rows of AR^{-1}, which provides an estimate of the ℓ_p norms of those rows, up to a factor of $d^{|1/2-1/p|}$. We uses these norm estimates, *e.g.*, as in the above algorithms or in the sampling algorithm of [5]. As discussed for the running time bound of [5], Theorem 7, this algorithm samples a number of rows proportional to $\bar{\kappa}_p^p(AR^{-1})d$. This factor, together with a sample complexity increase of $(d^{|1/2-1/p|})^p = d^{|p/2-1|}$ needed to compensate for error due to using Π_2, gives a sample complexity increase for the FastLpRegression algorithm while the leading term in the complexity (for $n \gg d$) is reduced from $O(nd^5 \log n)$ to $O(nd \log n)$. We modify Theorem 7 of [5] to obtain the following theorem.

THEOREM 5.4. *Given $\varepsilon \in (0, 1)$, $A \in \mathbb{R}^{n\times d}$, and $b \in \mathbb{R}^n$, there is a random sampling algorithm (the FastLpRegression algorithm described above) for ℓ_p regression that constructs a coreset specified by a diagonal sampling matrix D, and a solution vector $\hat{x} \in \mathbb{R}^d$ that minimizes the weighted regression objective $\|D(Ax - b)\|_p$. The solution $\hat{x}$ satisfies, with probability at least $1/2$, the relative error bound that $\|A\hat{x} - b\|_p \le (1 + \varepsilon)\|Ax - b\|_p$ for all $x \in \mathbb{R}^d$. Further, with probability $1 - o(1)$, the entire algorithm to construct $\hat{x}$ runs in time*

$$O\left(nd \log n + \phi_p(s, d)\right)=O\left(nd \log n + \tfrac{1}{\varepsilon^2} \operatorname{poly}(d, \log \tfrac{d}{\varepsilon})\right),$$

where $s = O(\varepsilon^{-2} d^k \log(1/\varepsilon))$ with $k = p + 1 + 4|p/2 - 1| + \max\{p/2, 1\}$, and $\phi_p(s, d)$ is the time to solve an ℓ_p regression problem on s vectors in d dimensions.

6 Numerical Implementation and Empirical Evaluation

In this section, we describe the results of our empirical evaluation. We have implemented and evaluated the Fast Cauchy Transforms (both FCT1 and FCT2) as well as the Cauchy transform (CT) of [13]. For completeness, we have also compared our method against two ℓ_2-based transforms: the Gaussian Transform (GT) and a version of the FJLT. Ideally, the evaluation would be based on the evaluating the distortion of the embedding, *i.e.*, evaluating the smallest κ such that

$$\|Ax\|_1 \le \|\Pi Ax\|_1 \le \kappa\|Ax\|_1, \quad \forall x \in \mathbb{R}^d,$$

where $\Pi \in \mathbb{R}^{r \times n}$ is one of the Cauchy transforms. Due to the non-convexity, there seems not to be a way to compute, tractably and accurately, the value of this κ. Instead, we evaluated both ℓ_1-based transforms (CT, FCT1, and FCT2) and ℓ_2-based transforms (GT and FJLT) based on how they perform in computing well-conditioned bases and approximating ℓ_1 regression problems.

In more detail, we have evaluated the quality of ℓ_1 conditioning using both ℓ_1-based transforms (CT, FCT1, and FCT2) and ℓ_2-based transforms (GT and FJLT) on a suite of matrices designed to illustrate the strengths and weaknesses of the respective methods; and we have evaluated the performance of these embedding methods on both small-scale and large-scale ℓ_1 regression problems. A full discussion of our empirical evaluation may be found in the technical report version of this paper [4]; here we describe only the evaluation on a large-scale ℓ_1 regression problem.

The ℓ_1 regression problem we consider is one with imbalanced and corrupted measurements. It is of size $5.24e9 \times 15$, and it is generated in the following way.

1. The true signal x^* is a standard Gaussian vector.

2. Each row of the design matrix A is a canonical vector, which means that we only estimate a single entry of x^* in each measurement. The number of measurements on the i-th entry of x^* is twice as large as that on the $(i+1)$-th entry, $i = 1, \ldots, 14$. We have 2.62 billion measurements on the first entry while only 0.16 million measurements on the last. Imbalanced measurements apparently create difficulties for sampling-based algorithms.

3. The response vector is given by

$$b_i = \begin{cases} 1000\varepsilon_i & \text{with probability } 0.001 \\ a_i^T x^* + \varepsilon_i & \text{otherwise} \end{cases}, i = 1, \ldots,$$

   where a_i is the i-th row of A and $\{\varepsilon_i\}$ are i.i.d. samples drawn from the Laplace distribution. 0.1% measurements are corrupted to simulate noisy real-world data. Due to these corrupted measurements, ℓ_2 regression won't give us accurate estimate, and ℓ_1 regression is certainly a more robust alternative.

Since the problem is separable, we know that an optimal solution is simply given by the median of responses corresponding to each entry.

Our empirical evaluation was performed on a Hadoop cluster with 40 cores. We implemented and compared Cauchy-conditioned sampling (CT), Gaussian-conditioned sampling (GT), un-conditioned sampling (NOCD), and uniform sampling (UNIF). Since

	$\frac{\|x-x^*\|_1}{\|x^*\|_1}$	$\frac{\|x-x^*\|_2}{\|x^*\|_2}$	$\frac{\|x-x^*\|_\infty}{\|x^*\|_\infty}$
CT	[0.008, 0.0115]	[0.00895, 0.0146]	[0.0113, 0.0211]
GT	[0.0126, 0.0168]	[0.0152, 0.0232]	[0.0184, 0.0366]
NOCD	[0.0823, 22.1]	[0.126, 70.8]	[0.193, 134]
UNIF	[0.0572, 0.0951]	[0.089, 0.166]	[0.129, 0.254]

Table 1: First and the third quartiles of relative errors in 1-, 2-, and ∞-norms. CT clearly performs the best. GT follows closely. NOCD generates large errors, while UNIF works but it is about a magnitude worse than CT.

A only has $2n$ non-zeros, CT takes $O(nd \log d)$ time instead of $O(nd^2 \log d)$, which makes it the fastest among CT, FCT1, and FCT2 on this particular problem. Moreover, even if A is dense, data at this scale are usually stored on secondary storage, and thus time spent on scanning the data typically dominates the overall running time. Therefore, we only implemented CT for this test. Note that the purpose of this test is not to compare CT, FCT1, and FCT2 (which we did in other parts of our empirical evaluation), but to reveal some inherent differences among ℓ_1 conditioned sampling (CT, FCT1, and FCT2), ℓ_2 conditioned sampling (GT and FJLT), and other sampling algorithms (NOCD and UNIF). For each algorithm, we sample approximately 100000 (0.019%) rows and repeat the sampling 100 times, resulting 100 approximate solutions. Note that those 100 approximate solutions can be computed simultaneously in a single pass.

We first check the overall performance of these sampling algorithms, measured by relative errors in 1-, 2-, and ∞-norms. The results are shown in Table 1. Since the algorithms are all randomized, we show the first and the third quartiles of the relative errors in 100 independent runs. We see that CT clearly performs the best, followed by GT. UNIF works but it is about a magnitude worse than CT. NOCD is close to UNIF at the first quartile, but makes very large errors at the third. Without conditioning, NOCD is more likely to sample outliers because the response from a corrupted measurement is much larger than that from a normal measurement. However, those corrupted measurements contain no information about x^*, which leads to NOCD's poor performance. UNIF treats all the measurements the same, but the measurements are imbalanced. Although we sample 100000 measurements, the expected number of measurements on the last entry is only 3.05, which downgrades UNIF's overall performance.

We continue to analyze entry-wise errors. Figure 4 draws the first and the third quartiles of entry-wise absolute errors, which clearly reveals the differences among ℓ_1 conditioned sampling, ℓ_2 conditioned sampling, and other sampling algorithms. While UNIF samples uni-

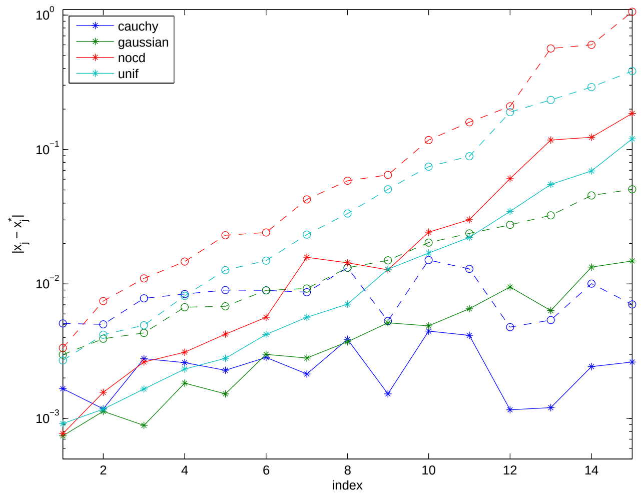

Figure 4: The first (solid) and the third (dashed) quartiles of entry-wise absolute errors for our large-scale ℓ_1 regression empirical evaluation. See the text for details.

formly row-wise, CT tends to sample uniformly entry-wise. Although not as good as other algorithms on the first entry, CT maintains the same error level across all the entries, delivering the best overall performance. The ℓ_2-based GT sits between CT and UNIF. ℓ_2 conditioning can help detect imbalanced measurements to a certain extent and adjust the sampling weights accordingly, but it is still biased towards the measurements on the first several entries.

To summarize, we have shown that ℓ_1 conditioned sampling indeed works on large-scale ℓ_1 regression problems and its performance looks promising. We obtained about two accurate digits (0.01 relative error) on a problem of size $5.24e9 \times 15$ by passing over the data twice and sampling only 100000 (0.019%) rows in a judicious manner.

7 Conclusion

We have introduced the Fast Cauchy Transform, an ℓ_1-based analog of fast Hadamard-based random projections. We have also demonstrated that this fast ℓ_1-based random projection can be used to develop algorithms with improved running times for a range of ℓ_1-based problems; we have provided extensions of these results to ℓ_p; and we have provided the first implementation and empirical evaluation of an ℓ_1-based random projection. Our empirical evaluation clearly demonstrates that for large and very rectangular problems, for which low-precision solutions are acceptable, our implementation follows our theory quite well; and it also points to interesting connections between ℓ_1-based projections and ℓ_2-based projections in practical settings. Understanding these connections theoretically, exploiting other properties such as sparsity, and using these ideas to develop improved algorithms for high-precision solutions to large-scale ℓ_1-based problems, are important future directions raised by our work.

References

[1] N. Ailon and B. Chazelle. The fast Johnson-Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

[2] J. P. Brooks and J. H. Dulá. The L1-norm best-fit hyperplane problem. *Applied Mathematics Letters*, 26(1):51–55, 2013.

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

[4] 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. Technical report. Preprint: arXiv:1207.4684 (2012).

[5] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M.W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM Journal on Computing*, (38):2060–2078, 2009.

[6] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[7] P. Drineas, M.W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2010.

[8] F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays presented to R. Courant on his 60th Birthday*, pages 187–204. 1948.

[9] L. Lovasz. *Algorithmic Theory of Numbers, Graphs, and Convexity.* CBMS-NSF Regional Conference Series in Applied Mathematics 50. SIAM, Philadelphia, 1986.

[10] M. W. Mahoney. *Randomized algorithms for matrices and data.* Foundations and Trends in Machine Learning. NOW Publishers, Boston, 2011. Also available at: arXiv:1104.5557.

[11] M. W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proc. Natl. Acad. Sci. USA*, 106:697–702, 2009.

[12] A. Maurer. A bound on the deviation probability for sums of non-negative random variables. *Journal of Inequalities in Pure and Applied Mathematics*, 4(1):Article 15, 2003.

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