ICM 2002 · Vol. III · 1–3

Smoothed Analysis of Algorithms

Daniel A. Spielman* Shang-Hua Teng†

Abstract

Spielman and Teng [STOC '01] introduced the smoothed analysis of algorithms to provide a framework in which one could explain the success in practice of algorithms and heuristics that could not be understood through the traditional worst-case and average-case analyses. In this talk, we survey some of the smoothed analyses that have been performed.

2000 Mathematics Subject Classification: 65Y20, 68Q17, 68Q25, 90C05.
Keywords and Phrases: Smoothed Analysis, Simplex Method, Condition Number, Interior Point Method, Perceptron Algorithm.

1. Introduction

The most common theoretical approach to understanding the behavior of algorithms is worst-case analysis. In worst-case analysis, one proves a bound on the worst possible performance an algorithm can have. A triumph of the Algorithms community has been the proof that many algorithms have good worst-case performance—a strong guarantee that is desirable in many applications. However, there are many algorithms that work exceedingly well in practice, but which are known to perform poorly in the worst-case or lack good worst-case analyses. In an attempt to rectify this discrepancy between theoretical analysis and observed performance, researchers introduced the average-case analysis of algorithms. In average-case analysis, one bounds the expected performance of an algorithm on random inputs. While a proof of good average-case performance provides evidence that an algorithm may perform well in practice, it can rarely be understood to explain the good behavior of an algorithm in practice. A bound on the performance of an algorithm under one distribution says little about its performance under another distribution, and may say little about the inputs that occur in practice. Smoothed analysis is a hybrid of worst-case and average-case analyses that inherits advantages of both.

*Department of Mathematics, M.I.T.
†Department of Computer Science, Boston University

In the formulation of smoothed analysis used in [27], we measure the maximum over inputs of the expected running time of a simplex algorithm under slight random perturbations of those inputs. To see how this measure compares with worst-case and average-case analysis, let X_n denote the space of linear-programming problems of length n and let $\mathcal{T}(x)$ denote the running time of the simplex algorithm on input x. Then, the worst-case complexity of the simplex algorithm is the function

$$\mathcal{C}_{worst}(n) = \max_{x \in X_n} \mathcal{T}(x),$$

and the average-case complexity of the algorithm is

$$\mathcal{C}_{ave}(n) = \mathbf{E}_{r \in X_n} \mathcal{T}(r),$$

under some suitable distribution on X_n. In contrast, the smoothed complexity of the simplex algorithm is the function

$$\mathcal{C}_{smooth}(n, \sigma) = \max_{x} \mathbf{E}_{r \in X_n} \mathcal{T}(x + \sigma \|x\| r),$$

where r is chosen according to some distribution, such as a Gaussian. In this case $\sigma \|x\| r$ is a Gaussian random vector of standard deviation $\sigma \|x\|$. We multiply by $\|x\|$ so that we can relate the magnitude of the perturbation to the magnitude of that which it perturbs.

In the smoothed analysis of algorithms, we measure the expected performance of algorithms under slight random perturbations of worst-case inputs. More formally, we consider the maximum over inputs of the expected performance of algorithms under slight random perturbations of those inputs. We then express this expectation as a function of the input size and the magnitude of the perturbation. While an algorithm with a good worst-case analysis will perform well on all inputs, an algorithm with a good smoothed analysis will perform well on almost all inputs in every small neighborhood of inputs. Smoothed analysis makes sense for algorithms whose inputs are subject to slight amounts of noise in their low-order digits, which is typically the case if they are derived from measurements of real-world phenomena. If an algorithm takes such inputs and has a good smoothed analysis, then it is unlikely that it will encounter an input on which it performs poorly. The name "smoothed analysis" comes from the observation that if one considers the running time of an algorithm as a function from inputs to time, then the smoothed complexity of the algorithm is the highest peak in the plot of this function after it is convolved with a small Gaussian.

In our paper introducing smoothed analysis, we proved that the simplex method has polynomial smoothed complexity [27]. The simplex method, which has been the most popular method of solving linear programs since the late 1940's, is the canonical example of a practically useful algorithm that could not be understood theoretically. While it was known to work very well in practice, contrived examples on which it performed poorly proved that it had horrible worst-case complexity[19, 20, 14, 13, 4, 17, 3]. The average-case complexity of the simplex method was proved to be polynomial [8, 7, 25, 16, 1, 2, 28], but this result was not considered to explain the performance of the algorithm in practice.

2. The Simplex Method for Linear Programming

We recall that a linear programming problem can be written in the form

$$\begin{aligned} &\text{maximize} && x^T c \\ &\text{subject to} && x^T a_i \leq b_i, \text{ for } 1 \leq i \leq n, \end{aligned} \tag{2.1}$$

where $c \in \mathbb{R}^d$, $a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$, for $1 \leq i \leq n$ In [27], we bound the smoothed complexity of a particular two-phase simplex method that uses the shadow-vertex pivot rule to solve linear programs in this form.

We recall that the constraints of the linear program, that $x^T a_i \leq b_i$, confine x to a (possibly open) polytope, and that the solution to the linear program is a vertex of this polytope. Simplex methods work by first finding some vertex of the polytope, and then walking along the 1-faces of the polytope from vertex to vertex, improving the objective function at each step. The pivot rule of a simplex algorithm dictates which vertex the algorithm should walk to when it has many to choose from. The shadow-vertex method is inspired by the simplicity of the simplex method in two-dimensions: in two-dimensions, the polytope is a polygon and the choice of next vertex is always unique. To lift this simplicity to higher dimensions, the shadow-vertex simplex method considers the orthogonal projection of the polytope defined by the constraints onto a two-dimensional space. The method then walks along the vertices of the polytope that are the pre-images of the vertices of the shadow polygon. By taking the appropriate shadow, it is possible to guarantee that the vertex optimizing the objective function will be encountered during this walk. Thus, the running time of the algorithm may be bounded by the number of vertices lying on the shadow polygon. Our first step in proving a bound on this number is a smoothed analysis of the number of vertices in a shadow. For example, we prove the bound:

Theorem 2.1 (Shadow Size) *Let $d \geq 3$ and $n > d$. Let c and t be independent vectors in $\mathbb{R}^d$, and let $a_1, \ldots, a_n$ be Gaussian random vectors in $\mathbb{R}^d$ of variance $\sigma^2 \leq \frac{1}{9d\log n}$ centered at points each of norm at most 1. Then, the expected number of vertices of the shadow polygon formed by the projection of $\{x : x^T a_i \leq 1\}$ onto* **Span** (t, c) *is at most*

$$\frac{58{,}888{,}678\ nd^3}{\sigma^6}. \tag{2.2}$$

This bound does not immediately lead to a bound on the running time of a shadow-vertex method as it assumes that t and c are fixed before the a_is are chosen, while in a simplex method the plane on which the shadow is followed depends upon the a_is. However, we are able to use the shadow size bound as a black-box to prove for a particular randomized two-phase shadow vertex simplex method:

Theorem 2.2 (Simplex Method) *Let $d \geq 3$ and $n \geq d+1$. Let $c \in \mathbb{R}^d$ and $b \in \{-1, 1\}^n$. Let $a_1, \ldots, a_n$ be Gaussian random vectors in $\mathbb{R}^d$ of variance $\sigma^2 \leq \frac{1}{9d\log n}$ centered at points each of norm at most 1. Then the expected number of simplex steps taken by the two-phase shadow-vertex simplex algorithm to solve the program specified by b, c, and $a_1, \ldots, a_n$ is at most*

$$(nd/\sigma)^{O(1)},$$

where the expectation is over the choice of $a_1, \ldots, a_n$ *and the random choices made by the algorithm.*

While the proofs of Theorems 2.1 and 2.2 are quite involved, we can provide the reader with this intuition for Theorem 2.1: *after perturbation, most of the vertices of the polytope defined by the linear program have an angle bounded away from flat.* This statement is not completely precise because "most" should be interpreted under a measure related to the chance a vertex appears in the shadow, as opposed to counting the number of vertices. Also, there are many ways of measuring high-dimensional angles, and different approaches are used in different parts of the proof. However, this intuitive statement tells us that most vertices on the shadow polygon should have angle bounded away from flat, which means that there cannot be too many of them.

One way in which angles of vertices are measured is by the condition number of their defining equations. A vertex of the polytope is given by a set of equations of the form

$$Cx = b.$$

The condition number of C is defined to be

$$\kappa(C) = \|C\| \left\|C^{-1}\right\|,$$

where we recall that

$$\|C\| = \max_x \frac{\|Cx\|}{\|x\|}$$

and that

$$\left\|C^{-1}\right\| = \min_x \frac{\|Cx\|}{\|x\|}.$$

The condition number is a measure of the sensitivity of x to changes in C and b, and is also a normalized measure of the distance of C to the set of singular matrices. For more information on the condition number of a matrix, we refer the reader to one of [15, 29, 10]. Condition numbers play a fundamental role in Numerical Analysis, which we will now discuss.

3. Smoothed Complexity Framework for Numerical Analysis

The condition number of a problem instance is generally defined to be the sensitivity of the output to slight perturbations of the problem instance. In Numerical Analysis, one often bounds the running time of an iterative algorithm in terms of the condition number of its input. Classical examples of algorithms subject to such analyses include Newton's method for root finding and the conjugate gradient method of solving systems of linear equations. For example, the number of iterations taken by the method of conjugate gradients is proportional to the square root of the condition number. Similarly, the running times of interior-point methods have been bounded in terms of condition numbers [22].

Blum [6] suggested that a complexity theory of numerical algorithms should be parameterized by the condition number of an input in addition to the input size. Smale [26] proposed a complexity theory of numerical algorithms in which one:

1. *proves a bound on the running time of an algorithm solving a problem in terms of its condition number, and then*
2. *proves that it is unlikely that a random problem instance has large condition number.*

This program is analogous to the average-case complexity of Theoretical Computer Science and hence shares the same shortcoming in modeling practical performance of numerical algorithms.

To better model the inputs that occur in practice, we propose replacing step 2 of Smale's program with

2'. *prove that for every input instance it is unlikely that a slight random perturbation of that instance has large condition number.*

That is, we propose to bound the smoothed value of the condition number. In contrast with the average-case analysis of condition numbers, our analysis can be interpreted as demonstrating that if there is a little bit of imprecision or noise in the input, then it is unlikely it is ill-conditioned. The combination of step 2' with step 1 of Smale's program provides a simple framework for performing smoothed analysis of numerical algorithms whose running time can be bounded by the condition number of the input.

4. Condition Numbers of Matrices

One of the most fundamental condition numbers is the condition number of matrices defined at the end of Section 2. In his paper, "The probability that a numerical analysis problem is difficult", Demmel [9] proved that it is unlikely that a Gaussian random matrix centered at the origin has large condition number. Demmel's bounds on the condition number were improved by Edelman [12]. As bounds on the norm of a random matrix are standard, we focus on the norm of the inverse, for which Edelman proved:

Theorem 4.1 (Edelman) *Let G be a d-by-d matrix of independent Gaussian random variables of variance 1 and mean 0. Then,*

$$\mathbf{Pr}\left[\left\|G^{-1}\right\| > t\right] \leq \frac{\sqrt{d}}{t}.$$

We obtain a smoothed analogue of this bound in work with Sankar [24]. That is, we show that for every matrix it is unlikely that the slight perturbation of that matrix has large condition number. The key technical statement is:

Theorem 4.2 (Sankar-Spielman-Teng) *Let $\bar{A}$ be an arbitrary d-by-d Real matrix and A a matrix of independent Gaussian random variables centered at $\bar{A}$, each of variance σ^2. Then*

$$\mathbf{Pr}\left[\left\|A^{-1}\right\| > x\right] < 1.823\frac{\sqrt{d}}{x\sigma}$$

In contrast with the techniques used by Demmel and Edelman, the techniques used in the proof of Theorem 4.2 are geometric and completely elementary. We now give the reader a taste of these techniques by proving the simpler:

Theorem 4.3 *Let $\bar{A}$ be an arbitrary d-by-d Real matrix and A a matrix of independent Gaussian random variables centered at $\bar{A}$, each of variance σ^2. Then,*

$$\mathbf{Pr}\left[\left\|A^{-1}\right\| \geq x\right] \leq d^{3/2}/x\sigma.$$

The first step of the proof is to relate $\left\|A^{-1}\right\|$ to a geometric quantity of the vectors in the matrix A. The second step is to bound the probability of a configuration under which this geometric quantity is small. The geometric quantity is given by:

Definition *For d vectors in $\mathbb{R}^d$, $a_1, \ldots, a_d$, define*

$$\mathbf{height}\left(a_1, \ldots, a_d\right) = \min_i \mathbf{dist}\left(a_i, \mathbf{Span}\left(a_1, \ldots, \hat{a}_i, \ldots, a_d\right)\right).$$

Lemma 4.5 *For d vectors in $\mathbb{R}^d$, $a_1, \ldots, a_d$,*

$$\left\|\left(a_1, \ldots, a_d\right)^{-1}\right\| \leq \sqrt{d}/\mathbf{height}\left(A\right).$$

Proof. Let t be a unit vector such that

$$\left\|\sum_{i=1}^{d} t_i a_i\right\| = 1/\left\|\left(a_1, \ldots, a_d\right)^{-1}\right\|.$$

Without loss of generality, let t_1 be the largest entry of t in absolute value, so $|t_1| \geq 1/\sqrt{d}$. Then, we have

$$\begin{aligned} \left\|a_1 + \sum_{i=2}^{d}(t_i/t_1)a_i\right\| &\leq \sqrt{n}/\left\|\left(a_1, \ldots, a_d\right)^{-1}\right\| \\ \implies \mathbf{dist}\left(a_1, \mathbf{Span}\left(a_2, \ldots, a_d\right)\right) &\leq \sqrt{d}/\left\|\left(a_1, \ldots, a_d\right)^{-1}\right\|. \qquad \spadesuit \end{aligned}$$

Proof of Theorem 4.3. Let $a_1, \ldots, a_d$ denote the columns of A. Lemma 4.5 tells us that if $\left\|A^{-1}\right\| \geq x$, then $\mathbf{height}\left(a_1, \ldots, a_d\right)$ is less than $\sqrt{d}/x$. For each i, the probability that the height of a_i above $\mathbf{Span}\left(a_1, \ldots, \hat{a}_i, \ldots, a_d\right)$ is less than $\sqrt{d}/x$ is at most

$$\sqrt{d}/x\sigma.$$

Thus,

$$\begin{aligned} &\mathbf{Pr}\left[\mathbf{height}\left(a_1, \ldots, a_d\right) < \sqrt{d}/x\right] \\ &\quad\leq \mathbf{Pr}\left[\exists i : \mathbf{dist}\left(a_i, \mathbf{Span}\left(a_1, \ldots, \hat{a}_i, \ldots, a_n\right)\right) < \sqrt{d}/x\right] \\ &\quad< n^{3/2}/x\sigma; \end{aligned}$$

so,

$$\mathbf{Pr}\left[\left\|\left(a_1, \ldots, a_d\right)^{-1}\right\| > x\right] < d^{3/2}/x\sigma. \qquad \spadesuit$$

Conjecture 1 *Let $\bar{A}$ be an arbitrary d-by-d Real matrix and A a matrix of independent Gaussian random variables centered at $\bar{A}$, each of variance σ^2. Then*

$$\mathbf{Pr}\left[\left\|A^{-1}\right\| > x\right] < \frac{\sqrt{d}}{x\sigma}.$$

5. Smoothed Condition Numbers of Linear Programs

The Perceptron algorithm solves linear programs of the following simple form:

Given a set of points $a_1, \ldots, a_n$, find a vector x such that $\langle a_i|x\rangle > 0$ for all i, if one exists.

One can define the condition number of the Perceptron problem to be the reciprocal of the "wiggle room" of the input. That is, let $S = \{x : \langle a_i|x\rangle > 0, \forall i\}$ and

$$\nu(a_1, \ldots, a_n) = \max_{x \in S}\left(\min_i \frac{|\langle a_i|x\rangle|}{\|a_i\|\,\|x\|}\right).$$

Then, the condition number of Perceptron problem is defined to be $1/\nu(a_1, \ldots, a_n)$.

The Perceptron algorithm works as follows: (1) Initialize $x = \mathbf{0}$; (2) Select any a_i such that $\langle a_i|x\rangle \leq 0$ and set $x = x + a_i/\|a_i\|$; (3) while $x \notin S$, go back to step (2).

Using the following two lemmas, Blum and Dunagan [5] obtained a smoothed analysis of the Perceptron algorithm.

Theorem 5.1 (Block-Novikoff) *On input $a_1, \ldots, a_n$, the perceptron algorithm terminates in at most $1/(\nu(a_1, \ldots, a_n))^2$ iterations.*

Theorem 5.2 (Blum-Dunagan) *Let $a_1, \ldots, a_n$ be Gaussian random vectors in $\mathbb{R}^d$ of variance $\sigma^2 < 1/(2d)$ centered at points each of norm at most 1. Then,*

$$\mathbf{Pr}\left[\frac{1}{\nu(a_1, \ldots, a_n)} > t\right] \leq \frac{nd^{1.5}}{\sigma t} \log \frac{\sigma t}{d^{1.5}}.$$

Setting $t = \frac{nd^{1.5}\log(n/\delta)}{\delta\sigma}$, Blum and Dunagan concluded

Theorem 5.3 (Blum-Dunagan) *Let $a_1, \ldots, a_n$ be Gaussian random vectors in $\mathbb{R}^d$ of variance $\sigma^2 < 1/(2d)$ centered at points each of norm at most 1. Then, there exists a constant c such that the probability that the perceptron takes more than $\frac{cd^3n^2\log^2(n/\delta)}{\delta^2\sigma^2}$ iterations is at most δ.*

In his seminal work, Renegar [21, 22, 23] defines the condition of a linear program to be the normalized reciprocal of its distance to the set of ill-posed linear programs, where an ill-posed program is one that can be made both feasible and infeasible or bounded and unbounded by arbitrarily small changes to its constraints.

Renegar proved the following theorem.

Theorem 5.4 (Renegar) *There is an interior point method such that, on input a linear program specified by (A, b, c) and an $\epsilon > 0$, will terminate in*

$$O(\sqrt{n+d}\log(\kappa(A, b, c)/\epsilon)$$

iterations and return either a solution within ϵ of the optimal or a certificate that the linear program is infeasible or unbounded.

With Dunagan, we recently proved the following smoothed bound on the condition number of a linear program [11]:

Theorem 5.5 (Dunagan-Spielman-Teng) *For any $\sigma^2 < 1/(nd)$, let $A = (a_1, \ldots, a_n)$ be a set of Gaussian random vectors in $\mathbb{R}^d$ of variance σ^2 centered at points $\bar{a}_1, \ldots, \bar{a}_n$, let b be a Gaussian random vector in $\mathbb{R}^d$ of variance σ^2 centered at $\bar{b}$ and let c be a Gaussian random vector in $\mathbb{R}^n$ of variance σ^2 centered at $\bar{c}$ such that $\sum_{i=1}^n \|\bar{a}_i\|^2 + \|\bar{b}\|^2 + \|\bar{c}\|^2 \leq 1$. Then*

$$\mathbf{Pr}_{A,b,c}\left[C(A,b,c) > t\right] < \frac{2^{14} n^2 d^{3/2}}{\sigma^2 t} \log^2 \frac{2^{10} n^2 d^{3/2} t}{\sigma^2},$$

and hence

$$\mathbf{E}_{A,b,c}\left[\log C(A,b,c)\right] \leq 21 + 3\log(nd/\sigma).$$

Combining these two theorem, we have

Theorem 5.6 (Smoothed Complexity of Interior Point Methods) *Let σ and (A,b,c) be as given in Theorem 5.5, Then, Renegar's interior point method solves the linear program specified by (A,b,c) to within precision ϵ in expected*

$$O\left(\sqrt{n+d}(21 + 3\log(nd/\sigma\epsilon))\right)$$

iterations.

6. Two Open Problems

As the norm of the inverse of matrix is such a fundamental quantity, it is natural to ask how the norms of the inverses of the $\binom{n}{d}$ d-by-d square sub-matrices of a d-by-n matrix behave. Moreover, a crude bound on the probability that many of these are large is a dominant term in the analysis of complexity of the simplex method in [27]. The bound obtained in that paper is:

Lemma 6.1 *Let $a_1, \ldots, a_n$ be Gaussian random vectors in $\mathbb{R}^d$ of variance $\sigma^2 \leq 1/9d\log n$ centered at points of norm at most 1. For $I \in \binom{[n]}{d}$ a d-set, let X_I denote the indicator random variable that is 1 if*

$$\left\|[a_i : i \in I]^{-1}\right\| \geq \frac{\sigma^2}{8d^{3/2}n^7}.$$

Then,

$$\mathbf{Pr}_{a_1,\ldots,a_n}\left[\frac{d}{2}\sum_I X_I < \left\lceil \frac{n-d-1}{2} \right\rceil \binom{n}{d-1}\right] \geq 1 - n^{-d} - n^{-n+d-1} - n^{-2.9d+1}.$$

Clearly, one should be able to prove a much stronger bound than that stated here, and thereby improve the bounds on the smoothed complexity of the simplex method.

While much is known about the condition numbers of random matrices drawn from continuous distributions, much less is known about matrices drawn from discrete distributions. We conjecture:

Conjecture 2 *Let A be a d-by-d matrix of independently and uniformly chosen ± 1 entries. Then,*

$$\mathbf{Pr}\left[\left\|A^{-1}\right\| > t\right] \leq \frac{\sqrt{d}}{t} + \alpha^n,$$

for some absolute constant $\alpha < 1$.

We remark that the case $t = \infty$, when the matrix A is singular, follows from a theorem of Kahn, Komlos and Szemeredi [18].

References

[1] I. Adler, R. M. Karp, and R. Shamir. A simplex variant solving an m x d linear program in $O(min(m^2, d^2))$ expected number of pivot steps. *J. Complexity*, 3:372–387, 1987.

[2] Ilan Adler and Nimrod Megiddo. A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension. *Journal of the ACM*, 32(4):871–895, October 1985.

[3] Nina Amenta and Gunter Ziegler. Deformed products and maximal shadows of polytopes. In B. Chazelle, J.E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, number 223 in Contemporary Mathematics, pages 57–90. Amer. Math. Soc., 1999.

[4] David Avis and Vasek Chvátal. Notes on Bland's pivoting rule. In *Polyhedral Combinatorics*, volume 8 of *Math. Programming Study*, pages 24–34. 1978.

[5] Avrim Blum and John Dunagan. Smoothed analysis of the perceptron algorithm for linear programming. In *SODA '02*, pages 905–914, 2002.

[6] Lenore Blum. Lectures on a theory of computation and complexity over the reals (or an arbitrary ring). In Erica Jen, editor, *The Proceedings of the 1989 Complex Systems Summer School, Santa Fe, New Mexico*, volume 2, pages 1–47, June 1989.

[7] Karl Heinz Borgwardt. *Untersuchungen zur Asymptotik der mittleren Schrittzahl von Simplexverfahren in der linearen Optimierung.* PhD thesis, Universitat Kaiserslautern, 1977.

[8] Karl Heinz Borgwardt. *The Simplex Method: a probabilistic analysis.* Number 1 in Algorithms and Combinatorics. Springer-Verlag, 1980.

[9] James Demmel. The probability that a numerical analysis problem is difficult. *Math. Comp.*, pages 499–480, 1988.

[10] James Demmel. *Applied Numerical Linear Algebra.* SIAM, 1997.

[11] John Dunagan, Daniel A. Spielman, and Shang-Hua Teng. Smoothed analysis of Renegar's condition number for linear programming. Available at `http://math.mit.edu/~spielman/SmoothedAnalysis`, 2002.

[12] Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM J. Matrix Anal. Appl.*, 9(4):543–560, 1988.

[13] Donald Goldfarb. Worst case complexity of the shadow vertex simplex algorithm. Technical report, Columbia University, 1983.

[14] Donald Goldfarb and William T. Sit. Worst case behaviour of the steepest edge simplex method. *Discrete Applied Math*, 1:277–285, 1979.

[15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins Series in the Mathematical Sciences. The Johns Hopkins University Press and North Oxford Academic, Baltimore, MD, USA and Oxford, England, 1983.

[16] M. Haimovich. The simplex algorithm is very good ! : On the expected number of pivot steps and related properties of random linear programs. Technical report, Columbia University, April 1983.

[17] Robert G. Jeroslow. The simplex algorithm with the pivot rule of maximizing improvement criterion. *Discrete Math.*, 4:367–377, 1973.

[18] Jeff Kahn, Janos Komlos, and Endre Szemeredi. On the probability that a random +/- 1 matrix is singular. *Journal of the American Mathematical Society*, 1995.

[19] V. Klee and G. J. Minty. How good is the simplex algorithm ? In Shisha, O., editor, *Inequalities – III*, pages 159–175. Academic Press, 1972.

[20] K. G. Murty. Computational complexity of parametric linear programming. *Math. Programming*, 19:213–219, 1980.

[21] J. Renegar. Some perturbation theory for linear programming. *Math. Programming*, 65(1, Ser. A):73–91, 1994.

[22] J. Renegar. Incorporating condition measures into the complexity theory of linear programming. *SIAM J. Optim.*, 5(3):506–524, 1995.

[23] J. Renegar. Linear programming, complexity theory and elementary functional analysis. *Math. Programming*, 70(3, Ser. A):279–351, 1995.

[24] Arvind Sankar, Daniel A. Spielman, and Shang-Hua Teng. Smoothed analysis of the condition numbers and growth factors of matrices. Available at `http://math.mit.edu/~spielman/SmoothedAnalysis`, 2002.

[25] S. Smale. On the average number of steps in the simplex method of linear programming. *Mathematical Programming*, 27:241–262, 1983.

[26] Steve Smale. Complexity theory and numerical analysis. *Acta Numerica*, pages 523–551, 1997.

[27] Daniel Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing (STOC '01)*, pages 296–305, 2001. Full version available at `http://math.mit.edu/~spielman/SmoothedAnalysis`.

[28] M.J. Todd. Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems. *Mathematical Programming*, 35:173–192, 1986.

[29] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.