MIT Sloan School of Management

Working Paper 4337-02
January 2002

COMPUTATIONAL EXPERIENCE AND THE EXPLANATORY VALUE OF CONDITION NUMBERS FOR LINEAR OPTIMIZATION

Fernando Ordóñez and Robert M. Freund

Computational Experience and the Explanatory Value of Condition Numbers for Linear Optimization

Fernando Ordóñez* and Robert M. Freund†

January 30, 2002

Abstract

The modern theory of condition numbers for convex optimization problems was initially developed for convex problems in the following conic format:

$$(CP_d): \qquad z_* := \min_x \{c^t x \mid Ax - b \in C_Y, x \in C_X\} \ .$$

The condition number $C(d)$ for (CP_d) has been shown in theory to be connected to a wide variety of behavioral and computational characteristics of (CP_d), from sizes of optimal solutions to the complexity of algorithms for solving (CP_d). The goal of this paper is to develop some computational experience and test the practical relevance of condition numbers for linear optimization on problem instances that one might encounter in practice. We used the NETLIB suite of linear optimization problems as a test bed for condition number computation and analysis. Our computational results indicate that 72% of the NETLIB suite problem instances are ill-conditioned. However, after pre-processing heuristics are applied, only 19% of the post-processed problem instances are ill-conditioned, and $\log C(d)$ of the finitely-conditioned post-processed problems is fairly nicely distributed. We also show that the number of IPM iterations needed to solve the problems in the NETLIB suite varies roughly linearly (and monotonically) with $\log C(d)$ of the post-processed problem instances. Empirical evidence yields a positive linear relationship between IPM iterations and $\log C(d)$ for the post-processed problem instances, significant at the 95% confidence level. Furthermore, 42% of the variation in IPM iterations among the NETLIB suite problem instances is accounted for by $\log C(d)$ of the problem instances after pre-processing.

*MIT Operations Research Center, 77 Massachusetts Avenue, Bldg. E40-149, Cambridge, MA 02139, USA, email: fordon@mit.edu

†MIT Sloan School of Management, 50 Memorial Drive, Cambridge, MA 02142, USA, email: rfreund@mit.edu

1 Introduction

The modern theory of condition numbers for convex optimization problems was initially developed in [20] for problems in the following conic format:

$$(CP_d) \qquad \begin{array}{rll} z_* := \min & c^t x & \\ \text{s.t.} & Ax - b \in C_Y & \\ & x \in C_X \ , & \end{array} \tag{1}$$

where, for concreteness, we consider A to be an $m \times n$ real matrix, $b \in I\!R^m$, $c \in I\!R^n$, and $C_X \subseteq I\!R^n$, $C_Y \subseteq I\!R^m$ are closed convex cones, and the data of the problem is the array $d = (A, b, c)$. We assume that we are given norms $\|x\|$ and $\|y\|$ on $I\!R^n$ and $I\!R^m$, respectively, and let $\|A\|$ denote the usual operator norm; let $\|v\|_*$ denote the dual norm associated with the norm $\|w\|$ on $I\!R^n$ or $I\!R^m$. We define the norm of the data instance $d = (A, b, c)$ by $\|d\| := \max\{\|A\|, \|b\|, \|c\|_*\}$.

The theory of condition numbers for (CP_d) focuses on three quantities – $\rho_P(d), \rho_D(d)$, and $C(d)$, to bound various behavioral and computational measures of (CP_d). The quantity $\rho_P(d)$ is called the "distance to primal infeasibility" and is defined as:

$$\rho_P(d) := \inf\{\|\Delta d\| \mid X_{d+\Delta d} = \emptyset\} \ ,$$

where X_d denotes the feasible region of (CP_d):

$$X_d := \{x \in I\!R^n \mid Ax - b \in C_Y, x \in C_X\} \ .$$

The quantity $\rho_D(d)$ is called the "distance to dual infeasibility" for the conic dual (CD_d) of (CP_d):

$$(CD_d) \qquad \begin{array}{rll} z^* := \max & b^t y & \\ \text{s.t.} & c - A^t y \in C_X^* & \\ & y \in C_Y^* \ . & \end{array} \tag{2}$$

and is defined similarly to $\rho_P(d)$ but using the dual problem instead. The quantity $C(d)$ is called the "condition number" of the problem instance d and is a scale-invariant reciprocal of the smallest data perturbation Δd that will render the perturbed data instance either primal or dual infeasible:

$$C(d) := \frac{\|d\|}{\min\{\rho_P(d), \rho_D(d)\}} \ . \tag{3}$$

The quantities $\rho_P(d), \rho_D(d)$, and $C(d)$ have been shown in theory to be connected to a wide variety of behavioral characteristics of (CP_d) and its dual, including bounds on sizes of feasible solutions, bounds on sizes of optimal solutions, bounds on optimal objective values, bounds on the sizes and aspect ratios of inscribed balls in the feasible region, bounds on the rate of deformation of the feasible region under perturbation, bounds on changes in optimal objective values under perturbation, and numerical bounds related to the linear algebra computations of certain algorithms, see [20], [4], [3], [6], [7], [8], [25], [23], [26], [24], [17], [18]. In the context of interior-point methods for linear and semidefinite optimization, these same three condition numbers have also been shown to be connected to various quantities of interest regarding the central trajectory, see [14] and [15]. The connection of these condition numbers to the complexity of algorithms has been developed in [6], [7], [21], [1], and [2], and some of the references contained therein.

Given the theoretical importance of these many results, it is natural to ask what are typical values of condition numbers that arise in practice? Are such problems typically well- or ill-conditioned?, and how are the condition numbers of such problems distributed? We begin to answer these question in this paper, where we compute and analyze the condition numbers for the NETLIB suite of industrial and academic LP problems. We present computational results that indicate that 72% of the NETLIB suite of linear optimization problem instances are ill-conditioned. However, after routine preprocessing by CPLEX 7.1, we find that only 19% of post-processed problem instances in the NETLIB suite are ill-conditioned, and that $\log C(d)$ of the finitely-conditioned post-processed problems is fairly nicely distributed.

In the case of modern interior-point-method (IPM) algorithms for linear optimization, the number of IPM iterations needed to solve a linear optimization instance has been observed to vary from 10 to 100 iterations, over a huge range of problem sizes, see [11], for example. Using the condition-number model for complexity analysis, one can bound the IPM iterations by $O(\sqrt{n}\log(C(d) + \cdots)$ for linear optimization in standard form, where the other terms in the bound are of a more technical nature, see [21] for details. (Of course, the IPM algorithms that are used in practice are different from the IPM algorithms that are used in the development of the complexity theory.) A natural question to ask then is whether the observed variation in the number of IPM iterations (albeit already small) can be accounted for by the condition numbers of the problem instances? In Section 4.3, we show that the number of IPM iterations needed to solve the problems in the NETLIB suite varies roughly linearly (and monotonically) with $\log C(d)$ of the post-processed problem instances. A simple linear regression model of IPM iterations as the dependent variable and $\log C(d)$ as the independent variable yields a positive

linear relationship between IPM iterations and $\log C(d)$ for the post-processed problem instance, significant at the 95% confidence level, with $R^2 = 0.4258$. Therefore, over 42% of the variation in IPM iterations among the NETLIB suite problems is accounted for by $\log C(d)$ of the problem instances after pre-processing.

The organization of this paper is as follows. In Section 2, we lay the groundwork for the computation of condition numbers for the NETLIB suite. Section 3 describes our methodology for computing condition numbers. Section 4 contains the computational results, and Section 5 contains summary conclusions.

2 Linear Programming, Conic Format, and Ground-Set Format

In order to attempt to address the issues raised in the previous section about practical computational experience and the relevance of condition numbers, one can start by computing the condition numbers for a suitably representative set of linear optimization instances that arise in practice, such as the NETLIB suite of industrial and academic linear optimization problems, see [13]. Practical methods for computing (or approximately computing) condition numbers for convex optimization problems in conic format (CP_d) have been developed in [7] and [17], and such methods are relatively easy to implement. It would then seem to be a simple task to compute condition numbers for the NETLIB suite. However, it turns out that there is a subtle catch that gets in the way of this simple strategy, and in fact necessitates using an extension of the condition number theory just a bit, as we now explain.

Linear optimization problem instances arising in practice are typically conveyed in the following format:

$$\begin{array}{ll} \min_x & c^t x \\ \text{s.t.} & A_i x \leq b_i \ , i \in L \\ & A_i x = b_i \ , i \in E \\ & A_i x \geq b_i \ , i \in G \\ & x_j \geq l_j \ , \ j \in L_B \\ & x_j \leq u_j \ , \ j \in U_B \ , \end{array} \tag{4}$$

where the first three sets of inequalities/equalities are the "constraints" and the last two sets of inequalities are the lower and upper bound conditions, and where $L_B, U_B \subset \{1, \ldots, n\}$. (LP problems in practice might also contain range constraints of the form "$b_{i,l} \leq A_i x \leq b_{i,u}$" as well. We ignore this for now.) By defining C_Y to be an appropriate cartesian product of nonnegative halflines $I\!R_+$, nonpositive halflines $-I\!R_+$, and the origin

$\{0\}$, we can naturally consider the constraints to be in the conic format "$Ax - b \in C_Y$" where $C_Y \subset I\!R^m$ and $m = |L| + |E| + |G|$. However, for the lower and upper bounds on the variables, there are different ways to convert the problem into the required conic format for computation and analysis of condition numbers. One way is to convert the lower and upper bound constraints into ordinary constraints. Assuming for expository convenience that all original constraints are equality constraints and that all lower and upper bounds are finite, this conversion of (4) to conic format is:

$$\begin{array}{ll} P_1 : \min_x & c^t x \\ \text{s.t.} & Ax - b = 0 \\ & Ix - l \geq 0 \\ & Ix - u \leq 0 \ . \end{array}$$

whose data for this now-conic format is:

$$\bar{A} := \begin{pmatrix} A \\ I \\ I \end{pmatrix}, \bar{b} := \begin{pmatrix} b \\ l \\ u \end{pmatrix}, \bar{c} := c$$

with cones:

$$\bar{C}_Y := \{0\}^m \times I\!R^n_+ \times -I\!R^n_+ \quad \text{and} \quad \bar{C}_X := I\!R^n \ .$$

Another way to convert the problem to conic format is to replace the variables x with nonnegative variables $s := x - l$ and $t := u - x$, yielding:

$$\begin{array}{ll} P_2 : \min_{s,t} & c^t s + c^t l \\ \text{s.t.} & As - (b - Al) = 0 \\ & Is + It - (u - l) = 0 \\ & s, t \geq 0 \ , \end{array}$$

whose data for this now-conic format is:

$$\tilde{A} := \begin{pmatrix} A & 0 \\ I & I \end{pmatrix}, \tilde{b} := \begin{pmatrix} b - Al \\ u - l \end{pmatrix}, \tilde{c} := c$$

with cones:

$$\tilde{C}_Y := \{0\}^m \times \{0\}^n \quad \text{and} \quad \tilde{C}_X := I\!R^n_+ \times I\!R^n_+ \ .$$

	P_1	P_2
$\|d\|$	428	405
$\rho_P(d)$	0.24450	0.90909
$\rho_D(d)$	0.00250	1.00000
$C(d)$	171,200	445

Table 1: Condition Numbers for two different conic conversions of the same problem.

These two different conic versions of the same original problem have different data and different cones, and so will generically have different condition numbers. This is illustrated on the following elementary example:

$$\begin{array}{rll} P: \min_{x_1,x_2} & x_1 & \\ \text{s.t.} & x_1 + x_2 \geq 1 & \\ & 400x_1 + x_2 \leq 420 & \\ & 1 \leq x_1 \leq 5 & \\ & -1 \leq x_2\ . & \end{array}$$

Table 1 shows condition numbers for problem P under the two different conversion strategies of P_1 and P_2, using the L_∞-norm in the space of the variables and the L_1-norm in the space of the right-hand-side vector. (The method for computing these condition numbers is described in Section 3.) As Table 1 shows, the choice of the conversion strategy can have a very large impact on the resulting condition number, thereby calling into question the practical significance of performing such conversions to conic format.

2.1 Ground-Set Model Format

As a means to overcome this problem, we instead consider the linear optimization problem (4) as an instance of the "ground-set" model format for convex optimization:

$$\begin{array}{rrll} & z_*(d) = & \min & c^t x \\ (GP_d) & & \text{s.t.} & Ax - b \in C_Y \\ & & & x \in P, \end{array} \tag{5}$$

where P is called the ground-set; P is no longer required to be a cone, but instead can be any closed convex set (and obviously includes the conic format as a special case when $P = C_X$). We denote problem (5) as (GP_d) because the instance depends on the data triplet $d = (A, b, c)$. The set P and the cone C_Y remain fixed as part of the definition of

the problem, and are not considered to be part of the data. Many aspects of the theory of condition numbers for conic convex optimization have been extended to the more general ground-set model format, see [16]. We will treat linear optimization problems conveyed in the format (4) to be an instance of (GP_d) by setting the ground-set P to be defined by the lower and upper bounds:

$$P := \{x \mid x_j \geq l_j \text{ for } j \in L_B,\ x_j \leq u_j \text{ for } j \in U_B\}\ , \tag{6}$$

and by re-writing the other constraints in conic format as described earlier. But now notice the data d does not include the lower and upper bound data $l_j, j \in L_B$ and $u_j, j \in U_B$. This is somewhat advantageous since in many settings of linear optimization the lower and/or upper bounds on most variables are 0 or 1 or other scalars that are "fixed" and are not generally thought of as subject to modification. (Of course, there are other settings where keeping the lower and upper bounds fixed independent of the other constraints is not as natural.)

2.2 Definition of $C(d)$ for Ground-Set Format

The general set-up for the development of condition-number theory for the ground-set model format is developed in [16]. We review this material briefly here for completeness.

Let X_d denote the feasible region of (GP_d):

$$X_d := \{x \in I\!R^n \mid Ax - b \in C_Y, x \in P\}$$

and define the primal distance to infeasibility $\rho_P(d)$:

$$\rho_P(d) := \inf\{\|\Delta d\| \mid X_{d+\Delta d} = \emptyset\}\ ,$$

similar to the conic case. In order to state the Lagrange dual of (GP_d) we use the following definitions, which depend on the ground-set P.

Recall that a vector $r \in I\!R^n$ is a ray of P if there is a vector $x \in P$ such that for all $\theta \geq 0$ the vector $x + \theta r \in P$. Let R denote the set of rays of P. Since P is a closed convex set, the set of rays R is a closed convex cone.

Define

$$C_P := \{(x, t) \mid x \in tP, t > 0\}$$

and let C denote the closed convex cone

$$C := \mathrm{cl}C_P\ ,$$

where "cl$S$" denotes the closure of a set S. Then it is straightforward to show that

$$C = C_P \cup \{(\, r, 0\,) \mid r \in R\}$$

and that

$$\begin{aligned} C^* &:= \left\{(s,v) \mid s^t x + v \geq 0 \text{ for any } x \in P\right\} \\ &= \left\{(s,v) \mid \inf_{x \in P} s^t x \geq -v\right\}. \end{aligned}$$

The Lagrange dual of (GP_d) is:

$$(GD_d) \qquad \begin{array}{lll} z^*(d) = & \max_{y,v} & b^t y - v \\ & \text{s.t.} & (c - A^t y, v) \in C^* \\ & & y \in C_Y^* \ . \end{array} \tag{7}$$

Let Y_d denote the feasible region of the dual problem (GD_d):

$$Y_d := \{(y,v) \in I\!R^m \times I\!R \mid (c - A^t y, v) \in C^* \ , \ y \in C_Y^*\}$$

and define the dual distance to infeasibility $\rho_D(d)$:

$$\rho_D(d) := \inf\{\|\Delta d\| \mid Y_{d+\Delta d} = \emptyset\} \ .$$

The quantities $\rho_P(d), \rho_D(d)$ are shown in [16] to be connected to a variety of behavioral characteristics of (GP_d) and its dual, including sizes of feasible solutions, sizes of optimal solutions, optimal objective values, sizes of inscribed balls, deformation of the feasible region under perturbation, and the complexity of a variety of algorithms.

Let $\mathcal{F}$ denote the set of data instances d for which both (GP_d) and (GD_d) are feasible, these are well-posed data instances:

$$\mathcal{F} = \{\ d \mid X_d \neq \emptyset \text{ and } Y_d \neq \emptyset\}\,.$$

For $d \in \mathcal{F}$, the definition of condition number in the ground set model is identical to the definition in the conic case:

$$C(d) := \frac{\|d\|}{\min\{\rho_P(d), \rho_D(d)\}} \ ,$$

it is the scale invariant reciprocal of the distance to the set of data instances that are either primal or dual infeasible. For completeness we mention that for a data instance $d \in \mathcal{F}$, the quantity $\rho(d) := \min\{\rho_P(d), \rho_D(d)\}$ is the distance to ill-posedness.

3 Computation of $\rho_P(d)$, $\rho_D(d)$, and $C(d)$ via Convex Optimization

In this section we show how to compute $\rho_P(d)$ and $\rho_D(d)$ for linear optimization data instances $d = (A, b, c)$ of the ground-set model format, as well as how to estimate $\|d\|$ and $C(d)$. The methodology presented herein is an extension of the methodology for computing $\rho_P(d)$ and $\rho_D(d)$ developed in [7]. We will make the following choice of norms throughout this section and the rest of this paper:

Assumption 1 *The norm on the space of the x variables in $\mathbb{R}^n$ is the L_∞-norm, and the norm on the space of the right-hand-side vector in $\mathbb{R}^m$ is the L_1-norm.*

Using this choice of norms, we will show in this section how to compute $\rho(d)$ for linear optimization problems by solving $2n + 2m$ LPs of size roughly that of the original problem. As is discussed in [7], the complexity of computing $\rho(d)$ very much depends on the chosen norms, with the norms given in Assumption 1 being particularly appropriate for efficient computation. We begin our analysis with a seemingly innocuous proposition which will prove to be very useful.

Proposition 1 *Consider the problem:*

$$\begin{array}{rlll} z_1 = & \min_{v,w} & f(v,w) & \\ & s.t. & \|v\|_\infty = 1 & \\ & & (v,w) \in K\ , & \end{array} \tag{8}$$

where $v \in \mathbb{R}^k$, $w \in \mathbb{R}^l$, K is a closed convex cone in $\mathbb{R}^{k+l}$, and $f(\cdot) : \mathbb{R}^{k+l} \mapsto \mathbb{R}_+$ is positively homogeneous of degree one ($f(\alpha(v,w)) = |\alpha| f(v,w)$ for any $\alpha \in \mathbb{R}$ and $(v,w) \in \mathbb{R}^{k+l}$). Then problem (8) and (9) have the same optimal values, i.e., $z_1 = z_2$, where

$$\begin{array}{rccc} z_2 = & \min_{i,j} & \min_{v,w} f(v,w) & \\ & i \in \{1,\ldots,n\}, j \in \{-1,1\} & v_i = j & \\ & & (v,w) \in K\ . & \end{array} \tag{9}$$

Proof: Let (v^*, w^*) be an optimal solution of (8). Since $\|v^*\|_\infty = 1$, there exist $i^* \in \{1,\ldots,n\}$ and $j^* \in \{-1,1\}$ such that $v^*_{i^*} = j^*$. Therefore (v^*, w^*) is feasible for the inner problem in (9) for $i = i^*$ and $j = j^*$, and so $z_2 \le z_1$.

If (v^*, w^*) is an optimal solution of (9) with $i = i^*$ and $j = j^*$, then $\|v^*\|_\infty \ge 1$. If $\|v^*\|_\infty = 1$, the point (v^*, w^*) is feasible for (8) which means that $z_1 \le z_2$, completing the proof. Therefore, assume that $\|v^*\|_\infty > 1$, and consider the new point $(\tilde{v}, \tilde{w}) :=$

$\frac{1}{\|v^*\|_\infty}(v^*, w^*) \in K$. Then $(\tilde{v}, \tilde{w})$ is feasible for an inner problem in (9) for some $i = \hat{i} \neq i^*$ and $j = \hat{j}$, and so $z_2 \leq f(\tilde{v}, \tilde{w}) = f\left(\frac{1}{\|v^*\|_\infty}(v^*, w^*)\right) = \frac{1}{\|v^*\|_\infty} f(v^*, w^*) \leq z_2$, which now implies that $(\tilde{v}, \tilde{w})$ is also an optimal solution of (9). Since $\|\tilde{v}\|_\infty = 1$, the previous argument implies that $z_1 \leq z_2$, completing the proof. ■

3.1 Computing $\rho_P(d)$ and $\rho_D(d)$

The following theorem, which is proved in [16], characterizes $\rho_P(d)$ and $\rho_D(d)$ as the optimal solution values of certain optimization problems.

Theorem 1 (see [16]) *Suppose $d \in \mathcal{F}$, and that the norms are chosen as in Assumption 1. Then $\rho_P(d) = j_P(d)$ and $\rho_D(d) = j_D(d)$, where*

$$\begin{array}{rcll} j_P(d) = & \min_{y,s,v} & \max\left\{\|A^t y + s\|_1, |b^t y - v|\right\} \\ & \text{s.t.} & \|y\|_\infty = 1 \\ & & y \in C_Y^* \\ & & (s, v) \in C^* \ , \end{array} \tag{10}$$

and

$$\begin{array}{rcll} j_D(d) = & \min_{x,p,g} & \max\left\{\|Ax - p\|_1, |c^t x + g|\right\} \\ & \text{s.t.} & \|x\|_\infty = 1 \\ & & x \in R \\ & & p \in C_Y \\ & & g \geq 0 \ . \end{array} \tag{11}$$

Neither (10) nor (11) are convex problems. However, both (10) and (11) are of the form (8), and so we can invoke Proposition 1, and solve (10) and (11) using problem (9). From Proposition 1, we have:

$$\begin{array}{rcll} \rho_P(d) = & \min & \min_{y,s,v} & \max\left\{\|A^t y + s\|_1, |b^t y - v|\right\} \\ & i \in \{1, \ldots, m\}, j \in \{-1, 1\} & \text{s.t.} & y_i = j \\ & & & y \in C_Y^* \\ & & & (s, v) \in C^* \end{array} \tag{12}$$

and

$$\begin{array}{rcll} \rho_D(d) = & \min & \min_{x,p,g} & \max\left\{\|Ax - p\|_1, |c^t x + g|\right\} \\ & i \in \{1, \ldots, n\}, j \in \{-1, 1\} & \text{s.t.} & x_i = j \\ & & & x \in R \\ & & & p \in C_Y \\ & & & g \geq 0. \end{array} \tag{13}$$

Taken together, (12) and (13) show that we can compute $\rho_P(d)$ by solving $2m$ convex optimization problems, and we can compute $\rho_D(d)$ by solving $2n$ convex optimization problems. In conclusion, we can compute $\rho(d)$ by solving $2n + 2m$ convex optimization problems, where all of the optimization problems involved are of the roughly the same size as the original problem GP_d.

Of course, each of the $2n + 2m$ convex problems in (12) and (13) will be computationally tractable only if we can conveniently work with the cones involved; we now show that for the special case of linear optimization models (4), there are convenient linear inequality characterizations of all of the cones involved in (12) and (13). The cone C_Y is easily seen to be:

$$C_Y = \{p \in I\!R^m \mid p_i \leq 0 \text{ for } i \in L, p_i = 0 \text{ for } i \in E, p_i \geq 0 \text{ for } i \in G\} \ , \tag{14}$$

and so

$$C_Y^* = \{y \in I\!R^m \mid y_i \leq 0 \text{ for } i \in L, y_i \in I\!R \text{ for } i \in E, y_i \geq 0 \text{ for } i \in G\} \ . \tag{15}$$

With the ground-set P defined in (6), we have:

$$R = \{x \in I\!R^n \mid x_j \geq 0 \text{ for } j \in L_B, \ x_j \leq 0 \text{ for } j \in U_B\} \ , \tag{16}$$

and also

$$C = \{(x,t) \in I\!R^n \times I\!R \mid t \geq 0, x_j \geq l_j t \text{ for } j \in L_B, \ x_j \leq u_j t \text{ for } j \in U_B\} \ . \tag{17}$$

The only cone whose characterization is less than obvious is C^*, which we now characterize. Consider the following system of linear inequalities in the variables $(s, v, s^+, s^-) \in I\!R^n \times I\!R \times I\!R^n \times I\!R^n$:

$$\begin{array}{rcll} s - s^+ + s^- & = & 0 & \\ s^+ & \geq & 0 & \\ s^- & \geq & 0 & \\ s_j^- & = & 0 & \text{for } j \in N \setminus U_B \\ s_j^+ & = & 0 & \text{for } j \in N \setminus L_B \\ v + \sum_{j \in L_B} l_j s_j^+ - \sum_{j \in U_B} u_j s_j^- & \geq & 0 \ , & \end{array} \tag{18}$$

where we use the notation $N := \{1, \ldots, n\}$ and $S \setminus T$ is the set difference $\{k \mid k \in S, k \notin T\}$.

Proposition 2 *For the ground-set P defined in (6), the cone C^* is characterized by*

$$C^* = \{(s,v) \in I\!R^n \times I\!R \mid (s, v, s^+, s^-) \text{ satisfies (18) for some } s^+, s^- \in I\!R^n\} \ .$$

Proof: Suppose first that (s, v) together with some s^+, s^- satisfies (18). Then for all $(x, t) \in C$ we have

$$\begin{aligned}(x,t)^t(s,v) &= \sum_{j\in L_B} s_j^+ x_j - \sum_{j\in U_B} s_j^- x_j + tv \\ &\geq \sum_{j\in L_B} s_j^+ l_j t - \sum_{j\in U_B} s_j^- u_j t + tv \qquad (19)\\ &\geq 0\ ,\end{aligned}$$

and so $(s, v) \in C^*$. Conversely, suppose that $(s, v) \in C^*$. Then

$$\begin{aligned}-\infty < -v \leq \min_{x\in P} s^t x = \min\ & \sum_{j=1}^{n} s_j x_j \\ \text{s.t.}\ & x_j \geq l_j \text{ for } j \in L_B \\ & x_j \leq u_j \text{ for } j \in U_B\ ,\end{aligned} \qquad (20)$$

and define s^+ and s^- to be the positive and negative parts of s, respectively. Then $s = s^+ - s^-, s^+ \geq 0$, and $s^- \geq 0$, and (20) implies $s_j^+ = 0$ for $j \in N \setminus L_B$, $s_j^- = 0$ for $j \in N \setminus U_B$, as well as the last inequality of (18), whereby (s, v, s^+, s^-) satisfies all inequalities of (18). ■

Taken together, we can use (14), (15), (16), (17), and Proposition 2 to re-write the right-most minimization problems of (12) and (13) and obtain:

$$\begin{array}{rccll}\rho_P(d) = & \min & \min_{y,s^+,s^-,v} & \max\left\{\left\|A^t y + s^+ - s^-\right\|_1, |b^t y - v|\right\} & \\ & i \in \{1,\ldots,m\} & \text{s.t.} & y_i = j & \\ & j \in \{-1,1\} & & y_l \leq 0 \quad \text{for } l \in L & \\ & & & y_l \geq 0 \quad \text{for } l \in G & \\ & & & s_k^- = 0 \quad \text{for } k \in N \setminus U_B & (21)\\ & & & s_k^+ = 0 \quad \text{for } k \in N \setminus L_B & \\ & & & v + \sum_{k\in L_B} l_k s_k^+ - \sum_{k\in U_B} u_k s_k^- \geq 0 & \\ & & & s^+, s^- \geq 0 & \end{array}$$

and

$$\rho_D(d) = \begin{array}{c} \min \\ i \in \{1,\ldots,n\} \\ j \in \{-1,1\} \end{array} \begin{array}{c} \min_{x,p,g} \\ \text{s.t.} \end{array} \begin{array}{l} \max\left\{\|Ax-p\|_1, |c^t x + g|\right\} \\ x_i = j \\ x_k \geq 0 \quad \text{if } k \in L_B \\ x_k \leq 0 \quad \text{for } k \in U_B \\ p_l \leq 0 \quad \text{for } l \in L \\ p_l = 0 \quad \text{for } l \in E \\ p_l \geq 0 \quad \text{for } l \in G \\ g \geq 0\ , \end{array} \tag{22}$$

whose right-most objective functions can then easily be converted to linear optimization problems by standard techniques. This then shows that we can indeed compute $\rho_P(d), \rho_D(d)$, and $\rho(d)$ by solving $2n + 2m$ LPs, under the choice of norms given in Assumption 1.

3.2 Computing $\|d\|$

In order to compute the condition number, given by $C(d) := \|d\|/\rho(d)$, we must also compute $\|d\| = \max\{\|A\|, \|b\|, \|c\|_*\}$. Under Assumption 1 we have $\|b\| = \|b\|_1$ and $\|c\|_* = \|c\|_1$, which are both easy to compute. However, $\|A\|$ is the operator norm, and so $\|A\| := \|A\|_{\infty,1} := \max\{\|Ax\|_1 \mid \|x\|_\infty = 1\}$, whose computation is a hard combinatorial problem. We therefore will bound $\|A\|_{\infty,1}$ and hence $\|d\|$ from below and above, using the following elementary norm inequalities:

$$\max\left\{\|A\|_{1,1}, \|A\|_{2,2}, \|A\|_F, \|Ae\|_1, \|A\hat{x}\|_1\right\} \leq \|A\|_{\infty,1} \leq \max\left\{\|A\|_{L_1}, \sqrt{nm}\,\|A\|_{2,2}\right\},$$

where

$$\begin{array}{rcl} \|A\|_{1,1} & = & \max_{j=1,\ldots,n} \|A_{\bullet j}\|_1\ , \\ \|A\|_{2,2} & = & \sqrt{\lambda_{\max}(A^t A)}, \\ \|A\|_F & = & \sqrt{\sum_{i=1}^m \sum_{j=1}^n (A_{i,j})^2}, \\ \|A\|_{L_1} & = & \sum_{i=1}^m \sum_{j=1}^n |A_{i,j}|, \end{array}$$

$e := (1,\ldots,1)^t$, and $\hat{x}$ is defined using $\hat{x}_j = \text{sign}(A_{i^*,j})$, where $i^* = \text{argmax}_{i=1,\ldots,m} \|A_{i\bullet}\|_1$.

4 Computational Results on the NETLIB Suite of Linear Optimization Problems

4.1 Condition Numbers for the NETLIB Suite prior to Pre-Processing

We computed the distances to ill-posedness and condition numbers for the NETLIB suite of linear optimization problems, using the methodology developed in Section 3. The NETLIB suite is comprised of 98 linear optimization problem instances from diverse application areas, collected over a period of many years. While this suite does not contain any truly large problems by today's standards, it is arguably the best publicly available collection of practical LP problems. The sizes of the problem instances in the NETLIB suite range from 32 variables and 28 constraints to problems with roughly 7,000 variables and 3,000 constraints. 44 of the 98 problems in the suite have non-zero lower bound constraints and/or upper bound constraints on the variables, and five problems have range constraints. We omitted the five problems with range constraints (boeing1, boeing2, forplan, nesm, seba) for the purposes of our analysis. We also omitted an additional five problems in the suite because they are not readily available in MPS format (qap8, qap12, qap15, stocfor3, truss). (These problems are each presented as a code that can be used to generate an MPS file.) Of the remaining 88 problems, there were five more problems (dfl001, fit2p, maros-r7, pilot, pilot87) for which our methodology was unable to compute the distance to ill-posedness in spite of over a full week of computation time. These problems were omitted as well, yielding a final sample set of 83 linear optimization problems. The burden of computing the distances to ill-posedness for the NETLIB suite via the solution of $2n + 2m$ LPs obviously grows with the dimensions of the problem instances. On afiro, which is a small problem instance ($n = 28$, $m = 32$), the total computation time amounted to only 0.28 seconds of machine time, whereas for d6cube ($n = 5,442$ and $m = 402$ after pre-processing), the total computation time was $77,152.43$ seconds of machine time (21.43 hours).

Table 2 shows the distances to ill-posedness and the condition number estimates for the 83 problems, using the methodology for computing $\rho_P(d)$ and $\rho_D(d)$ and for estimating $\|d\|$ presented in Section 3. All linear programming computation was performed using CPLEX 7.1 (function *primopt*).

Table 2: Condition Numbers for the NETLIB Suite prior to Pre-Processing

Problem	$\rho_P(d)$	$\rho_D(d)$	$\|d\|$ Lower Bound	$\|d\|$ Upper Bound	$\log C(d)$ Lower Bound	$\log C(d)$ Upper Bound
25fv47	0.000000	0.000000	30,778	55,056	∞	∞
80bau3b	0.000000	0.000000	142,228	142,228	∞	∞
adlittle	0.000000	0.051651	68,721	68,721	∞	∞
afiro	0.397390	1.000000	1,814	1,814	3.7	3.7
agg	0.000000	0.771400	5.51E+07	5.51E+07	∞	∞
agg2	0.000000	0.771400	1.73E+07	1.73E+07	∞	∞
agg3	0.000000	0.771400	1.72E+07	1.72E+07	∞	∞
bandm	0.000000	0.000418	10,200	17,367	∞	∞
beaconfd	0.000000	0.000000	15,322	19,330	∞	∞
blend	0.003541	0.040726	1,020	1,255	5.5	5.5
bnl1	0.000000	0.106400	8,386	9,887	∞	∞
bnl2	0.000000	0.000000	36,729	36,729	∞	∞
bore3d	0.000000	0.003539	11,912	12,284	∞	∞
brandy	0.000000	0.000000	7,254	10,936	∞	∞
capri	0.000252	0.095510	33,326	33,326	8.1	8.1
cycle	0.000000	0.000000	365,572	391,214	∞	∞
czprob	0.000000	0.008807	328,374	328,374	∞	∞
d2q06c	0.000000	0.000000	171,033	381,438	∞	∞
d6cube	0.000000	2.000000	47,258	65,574	∞	∞
degen2	0.000000	1.000000	3,737	3,978	∞	∞
degen3	0.000000	1.000000	4,016	24,646	∞	∞
e226	0.000000	0.000000	22,743	37,344	∞	∞
etamacro	0.000000	0.200000	31,249	63,473	∞	∞
fffff800	0.000000	0.033046	1.55E+06	1.55E+06	∞	∞
finnis	0.000000	0.000000	31,978	31,978	∞	∞
fit1d	3.500000	∞	493,023	618,065	5.1	5.2
fit1p	1.271887	0.437500	218,080	384,121	5.7	5.9
fit2d	317.000000	∞	1.90E+06	2.25E+06	3.8	3.9
ganges	0.000000	1.000000	1.29E+06	1.29E+06	∞	∞
gfrd-pnc	0.000000	0.347032	1.63E+07	1.63E+07	∞	∞
greenbea	0.000000	0.000000	21,295	26,452	∞	∞
greenbeb	0.000000	0.000000	21,295	26,452	∞	∞
grow15	0.572842	0.968073	209	977	2.6	3.2
grow22	0.572842	0.968073	303	1,443	2.7	3.4
grow7	0.572842	0.968073	102	445	2.3	2.9
israel	0.027248	0.166850	2.22E+06	2.22E+06	7.9	7.9
kb2	0.000201	0.018802	10,999	11,544	7.7	7.8
lotfi	0.000306	0.000000	166,757	166,757	∞	∞

Problem	$\rho_P(d)$	$\rho_D(d)$	$\|d\|$ Lower Bound	$\|d\|$ Upper Bound	$\log C(d)$ Lower Bound	$\log C(d)$ Upper Bound
maros	0.000000	0.000000	2.51E+06	2.55E+06	∞	∞
modszk1	0.000000	0.108469	1.03E+06	1.03E+06	∞	∞
perold	0.000000	0.000943	703,824	2.64E+06	∞	∞
pilot.ja	0.000000	0.000750	2.67E+07	1.40E+08	∞	∞
pilot.we	0.000000	0.044874	5.71E+06	5.71E+06	∞	∞
pilot4	0.000000	0.000075	763,677	1.09E+06	∞	∞
pilotnov	0.000000	0.000750	2.36E+07	1.35E+08	∞	∞
recipe	0.000000	0.000000	14,881	19,445	∞	∞
sc105	0.000000	0.133484	3,000	3,000	∞	∞
sc205	0.000000	0.010023	5,700	5,700	∞	∞
sc50a	0.000000	0.562500	1,500	1,500	∞	∞
sc50b	0.000000	0.421875	1,500	1,500	∞	∞
scagr25	0.021077	0.034646	430,977	430,977	7.3	7.3
scagr7	0.022644	0.034646	120,177	120,177	6.7	6.7
scfxm1	0.000000	0.000000	21,425	22,816	∞	∞
scfxm2	0.000000	0.000000	44,153	45,638	∞	∞
scfxm3	0.000000	0.000000	66,882	68,459	∞	∞
scorpion	0.000000	0.949393	5,622	5,622	∞	∞
scrs8	0.000000	0.000000	68,630	69,449	∞	∞
scsd1	5.037757	1.000000	1,752	1,752	3.2	3.2
scsd6	1.603351	1.000000	2,973	2,973	3.5	3.5
scsd8	0.268363	1.000000	5,549	5,549	4.3	4.3
sctap1	0.032258	1.000000	8,240	17,042	5.4	5.7
sctap2	0.586563	1.000000	32,982	72,870	4.7	5.1
sctap3	0.381250	1.000000	38,637	87,615	5.0	5.4
share1b	0.000015	0.000751	60,851	87,988	9.6	9.8
share2b	0.001747	0.287893	19,413	23,885	7.0	7.1
shell	0.000000	1.777778	253,434	253,434	∞	∞
ship04l	0.000000	13.146000	811,956	811,956	∞	∞
ship04s	0.000000	13.146000	515,186	515,186	∞	∞
ship08l	0.000000	21.210000	1.91E+06	1.91E+06	∞	∞
ship08s	0.000000	21.210000	1.05E+06	1.05E+06	∞	∞
ship12l	0.000000	7.434000	794,932	794,932	∞	∞
ship12s	0.000000	7.434000	381,506	381,506	∞	∞
sierra	0.000000	∞	6.60E+06	6.61E+06	∞	∞
stair	0.000580	0.000000	976	1,679	∞	∞
standata	0.000000	1.000000	21,428	23,176	∞	∞
standgub	0.000000	0.000000	21,487	23,235	∞	∞
standmps	0.000000	1.000000	22,074	23,824	∞	∞
stocfor1	0.001203	0.011936	23,212	23,441	7.3	7.3
stocfor2	0.000437	0.000064	462,821	467,413	9.9	9.9

Problem	$\rho_P(d)$	$\rho_D(d)$	$\|d\|$ Lower Bound	$\|d\|$ Upper Bound	$\log C(d)$ Lower Bound	$\log C(d)$ Upper Bound
tuff	0.000000	0.017485	136,770	145,448	∞	∞
vtp.base	0.000000	0.500000	530,416	534,652	∞	∞
wood1p	0.000000	1.000000	3.66E+06	5.04E+06	∞	∞
woodw	0.000000	1.000000	9.86E+06	1.35E+07	∞	∞

Table 3 presents some summary statistics of the condition number computations from Table 2. As the table shows, 72% (60/83) of the problems in the NETLIB suite are ill-conditioned due to either $\rho_P(d) = 0$ or $\rho_D(d) = 0$ or both. Furthermore, notice that among these 60 ill-conditioned problems, that almost all of these (58 out of 60) have $\rho_P(d) = 0$. This means that for 70% (58/83) of the problems in the NETLIB suite, arbitrarily small changes in the data will render the primal problem infeasible.

Notice from Table 2 that there are three problems for which $\rho_D(d) = \infty$, namely fit1d, fit2d, and sierra. This can only happen when the ground-set P is bounded, which for linear optimization means that all variables have finite lower and upper bounds.

Table 3: Summary Statistics of Distances to Ill-Posedness for the NETLIB Suite prior to Pre-Processing.

		$\rho_D(d)$ 0	$\rho_D(d)$ Finite	$\rho_D(d)$ ∞	Totals
$\rho_P(d)$	0	18	39	1	58
	Finite	2	21	2	25
	∞	0	0	0	0
	Totals	20	60	3	83

4.2 Condition Numbers for the NETLIB Suite after Pre-Processing

Most commercial software packages for solving linear optimization problems perform pre-processing heuristics prior to solving a problem instance. These heuristics typically include row and/or column re-scaling, checks for linearly dependent equations, heuristics for identifying and eliminating redundant variable lower and upper bounds, etc. The original problem instance is converted to a post-processed instance by the processing heuristics, and it is this post-processed instance that is used as input to solution software. In CPLEX 7.1, the post-processed problem can be accessed using function *prslvwrite*. This function writes the post-processed problem to disk, from where it can be read.

In order to get a sense of the distribution of the condition numbers of the problems that are input to a modern IPM solver, we computed condition numbers for the post-processed versions of the 83 NETLIB suite problems, where the processing used was the default CPLEX preprocessing with the linear dependency check option activated. Table 4 shows the condition numbers in detail for the post-processed versions of the problems, and Table 5 presents some summary statistics of these condition numbers. Notice from Table 5 that only 19% (16/83) of the post-processed problems in the NETLIB suite are ill-posed. The pre-processing heuristics have increased the number of problems with finite condition numbers to 67 problems. In contrast to the original problems, the vast majority of post-processed problems have finite condition numbers.

Table 4: Condition Numbers for the NETLIB Suite after Pre-Processing by CPLEX 7.1

			$\|d\|$		$\log C(d)$	
Problem	$\rho_P(d)$	$\rho_D(d)$	Lower Bound	Upper Bound	Lower Bound	Upper Bound
25fv47	0.000707	0.000111	35,101	54,700	8.5	8.7
80bau3b	0.000000	0.000058	126,355	126,355	∞	∞
adlittle	0.004202	1.000488	68,627	68,627	7.2	7.2
afiro	0.397390	1.000000	424	424	3.0	3.0
agg	0.000000	0.031728	3.04E+07	3.04E+07	∞	∞
agg2	0.000643	1.005710	1.57E+07	1.57E+07	10.4	10.4
agg3	0.000687	1.005734	1.56E+07	1.56E+07	10.4	10.4
bandm	0.001716	0.000418	7,283	12,364	7.2	7.5
beaconfd	0.004222	1.000000	6,632	6,632	6.2	6.2
blend	0.011327	0.041390	872	1,052	4.9	5.0
bnl1	0.000016	0.159015	8,140	9,544	8.7	8.8
bnl2	0.000021	0.000088	18,421	20,843	8.9	9.0
bore3d	0.000180	0.012354	8,306	8,306	7.7	7.7
brandy	0.000342	0.364322	4,342	7,553	7.1	7.3
capri	0.000375	0.314398	30,323	30,323	7.9	7.9
cycle	0.000021	0.009666	309,894	336,316	10.2	10.2
czprob	0.000000	0.001570	206,138	206,138	∞	∞
d2q06c	0.000000	0.003925	172,131	378,209	∞	∞
d6cube	0.945491	2.000000	43,629	60,623	4.7	4.8
degen2	0.000000	1.000000	2,613	3,839	∞	∞
degen3	0.000000	1.000000	4,526	24,090	∞	∞
e226	0.000737	0.021294	21,673	35,518	7.5	7.7
etamacro	0.001292	0.200000	55,527	87,767	7.6	7.8
fffff800	0.000000	0.033046	696,788	696,788	∞	∞
finnis	0.000000	0.000000	74,386	74,386	∞	∞
fit1d	3.500000	∞	493,023	617,867	5.1	5.2
fit1p	1.389864	1.000000	218,242	383,871	5.3	5.6

Problem	$\rho_P(d)$	$\rho_D(d)$	$\|d\|$ Lower Bound	$\|d\|$ Upper Bound	$\log C(d)$ Lower Bound	$\log C(d)$ Upper Bound
fit2d	317.000000	∞	1.90E+06	2.24E+06	3.8	3.8
ganges	0.000310	1.000000	143,913	143,913	8.7	8.7
gfrd-pnc	0.015645	0.347032	1.22E+07	1.22E+07	8.9	8.9
greenbea	0.000033	0.000004	65,526	65,526	10.2	10.2
greenbeb	0.000034	0.000007	43,820	43,820	9.8	9.8
grow15	0.572842	0.968073	209	977	2.6	3.2
grow22	0.572842	0.968073	303	1,443	2.7	3.4
grow7	0.572842	0.968073	102	445	2.3	2.9
israel	0.135433	0.166846	2.22E+06	2.22E+06	7.2	7.2
kb2	0.000201	0.026835	10,914	11,054	7.7	7.7
lotfi	0.000849	0.001590	170,422	170,422	8.3	8.3
maros	0.000000	0.006534	1.76E+06	1.80E+06	∞	∞
modszk1	0.016030	0.114866	1.03E+06	1.03E+06	7.8	7.8
perold	0.000000	0.002212	1.56E+06	2.35E+06	∞	∞
pilot.ja	0.000000	0.001100	2.36E+07	1.36E+08	∞	∞
pilot.we	0.000000	0.044874	5.71E+06	5.71E+06	∞	∞
pilot4	0.000399	0.002600	696,761	1.03E+06	9.2	9.4
pilotnov	0.000000	0.001146	2.36E+07	1.32E+08	∞	∞
recipe	0.063414	0.000000	13,356	15,815	∞	∞
sc105	0.778739	0.400452	3,000	3,000	3.9	3.9
sc205	0.778739	0.030068	5,700	5,700	5.3	5.3
sc50a	0.780744	1.000000	1,500	1,500	3.3	3.3
sc50b	0.695364	1.000000	1,500	1,500	3.3	3.3
scagr25	0.021191	0.049075	199,859	199,859	7.0	7.0
scagr7	0.022786	0.049075	61,259	61,259	6.4	6.4
scfxm1	0.000010	0.002439	20,426	21,811	9.3	9.3
scfxm2	0.000010	0.002439	38,863	43,630	9.6	9.6
scfxm3	0.000010	0.002439	57,300	65,449	9.8	9.8
scorpion	0.059731	0.995879	123,769	123,769	6.3	6.3
scrs8	0.009005	0.004389	66,362	68,659	7.2	7.2
scsd1	5.037757	1.000000	1,752	1,752	3.2	3.2
scsd6	1.603351	1.000000	2,973	2,973	3.5	3.5
scsd8	0.268363	1.000000	5,549	5,549	4.3	4.3
sctap1	0.032258	1.000000	7,204	15,186	5.3	5.7
sctap2	0.669540	1.000000	27,738	64,662	4.6	5.0
sctap3	0.500000	1.000000	32,697	78,415	4.8	5.2
share1b	0.000015	0.000751	1.67E+06	1.67E+06	11.0	11.0
share2b	0.001747	0.287893	19,410	23,882	7.0	7.1
shell	0.000263	0.253968	874,800	874,800	9.5	9.5
ship04l	0.000386	25.746000	881,005	881,005	9.4	9.4
ship04s	0.000557	25.746000	545,306	545,306	9.0	9.0

Problem	$\rho_P(d)$	$\rho_D(d)$	$\|d\|$ Lower Bound	$\|d\|$ Upper Bound	$\log C(d)$ Lower Bound	$\log C(d)$ Upper Bound
ship08l	0.000000	22.890000	1.57E+06	1.57E+06	∞	∞
ship08s	0.000000	22.890000	816,531	816,531	∞	∞
ship12l	0.000124	7.434000	748,238	748,238	9.8	9.8
ship12s	0.000149	7.434000	340,238	340,238	9.4	9.4
sierra	0.001039	47.190000	6.60E+06	6.61E+06	9.8	9.8
stair	0.003800	0.163162	7,071	7,071	6.3	6.3
standata	0.090909	1.000000	4,931	5,368	4.7	4.8
standgub	0.090909	1.000000	4,931	5,368	4.7	4.8
standmps	0.020000	1.000000	12,831	12,831	5.8	5.8
stocfor1	0.002130	0.109062	10,833	29,388	6.7	7.1
stocfor2	0.000811	0.000141	45,458	616,980	8.5	9.6
tuff	0.000025	0.047081	131,554	138,783	9.7	9.7
vtp.base	0.005287	3.698630	17,606	17,606	6.5	6.5
wood1p	0.059008	1.442564	2.11E+06	3.25E+06	7.6	7.7
woodw	0.009357	1.000000	5.68E+06	7.26E+06	8.8	8.9

Table 5: Summary Statistics of Distances to Ill-Posedness for the NETLIB Suite after Pre-Processing by CPLEX 7.1.

		$\rho_D(d)$ 0	$\rho_D(d)$ Finite	$\rho_D(d)$ ∞	Totals
$\rho_P(d)$	0	1	14	0	15
	Finite	1	65	2	68
	∞	0	0	0	0
	Totals	2	79	2	83

Figure 1 presents a histogram of the condition numbers of the post-processed problems taken from Table 4. The condition number of each problem is represented by the geometric mean of the upper and lower bound estimates in this histogram. The rightmost column in the figure is used to tally the number of problems for which $C(d) = \infty$, and is shown to give a more complete picture of the data. This histogram shows that of the problems with finite condition number, $\log C(d)$ is fairly nicely distributed between 2.6 and 11.0. Of course, when $C(d) = 10^{11}$, it is increasingly difficult to distinguish between a finite and non-finite condition number.

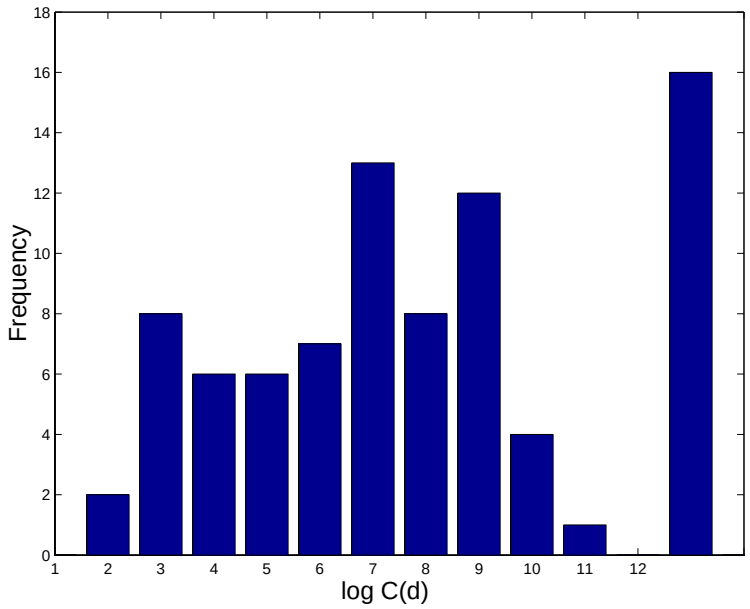

Figure 1: Histogram of Condition Numbers for the NETLIB Suite After Pre-Processing Heuristics were Applied (using the geometric mean of the lower and upper bound estimates of $C(d)$)

4.3 Condition Numbers and the Observed Performance of Interior-Point Methods on the NETLIB Suite

It is part of the folklore of linear optimization that the number of iterations of the simplex method tends to grow roughly linearly in the number of variables, see [22] for a survey of studies of simplex method computational performance. This observed linear growth is (fortunately) in stark contrast to the worst-case iteration bound for the simplex method, which is exponential in the dimension of the problem, see [9]. Of course, even here one must bear in mind that implemented versions of the simplex algorithm (on which computational performance is assessed) do not correspond to the theoretical versions of the simplex algorithm (on which theory and complexity analysis is developed) in many respects, including the way degeneracy is handled, feasibility checks are performed, numerical tolerances are used, etc.

In the case of modern IPM algorithms for linear optimization, the number of IPM iterations needed to solve a linear optimization instance has been observed to be fairly constant over a huge range of problem sizes; for the NETLIB suite the number of iterations varies between 8 and 48 using CPLEX 7.1 *baropt*; for other codes the numbers are a bit different. Extensive computational experience over the past 15 years has shown that the IPM iterations needed to solve a linear optimization problem instance vary in

the range between 10-100 iterations. There is some evidence that the number of IPM iterations grows roughly as $\log n$ on a particular class of structured problem instances, see for example [10].

The observed performance of modern IPM algorithms is fortunately superior to the worst-case bounds on IPM iterations that arise via theoretical complexity analysis. Depending on the complexity model used, one can bound the number of IPM iterations from above by $\sqrt{\vartheta}\tilde{L}$, where ϑ is the number of inequalities plus the number of variables with at least one bound in the problem instance:

$$\vartheta := |L| + |G| + |L_B| + |U_B| - |L_B \cap U_B| \ , \tag{23}$$

and $\tilde{L}$ is the bit-size of a binary encoding of the problem instance data, see [19] (subtraction of the final term of (23) is shown in [5]). The bit-size model was a motivating force for modern polynomial-time LP algorithms, but is viewed today as somewhat outdated in the context of linear and nonlinear optimization. Using instead the condition-number model for complexity analysis, one can bound the IPM iterations by $O(\sqrt{\vartheta}\log(C(d)+\cdots)$, where the other terms in the bound are of a more technical nature, see [21] for details. Similar to the case of the simplex algorithm, the IPM algorithms that are used in practice are different from the IPM algorithms that are used in the development of the complexity theory.

A natural question to ask is whether the observed variation in the number of IPM iterations (albeit already small) can be accounted for by the condition numbers of the problem instances? The finite condition numbers of the 67 post-processed problems from the NETLIB suite shown in Table 4 provide a rich set of data that can be used to explore this question. Here the goal is to assess whether or not condition numbers are relevant for understanding the practical performance of IPM algorithms (and is *not* aimed at validating the complexity theory).

In order to assess any relationship between condition numbers and IPM iterations for the NETLIB suite, we first solved and recorded the IPM iterations for the 83 problems from the NETLIB suite. The problems were pre-processed with the linear dependency check option and solved with CPLEX 7.1 function *baropt* with default parameters. The default settings use the standard barrier algorithm, include a starting heuristic that sets the initial dual solution to zero, and a convergence criteria of a relative complementarity smaller than 10^{-8}. The iteration counts are shown in Table 6. Notice that these iteration counts vary between 8 and 48.

Figure 2 shows a scatter plot of the number of IPM iterations taken by CPLEX 7.1 to solve the 83 problems in the NETLIB suite after pre-processing (from Table 6) and $\sqrt{\vartheta}\log C(d)$ of the post-processed problems (using the $\log C(d)$ estimates from columns 6 and 7 from Table 4). In the figure, the horizontal lines represent the range for $\sqrt{\vartheta}\log C(d)$ due to the lower and upper estimates of $C(d)$ from the last two columns of

Table 6: IPM Iterations for the NETLIB Suite using CPLEX 7.1 *baropt*

Problem	IPM Iterations	Problem	IPM Iterations	Problem	IPM Iterations
25fv47	22	ganges	13	scrs8	20
80bau3b	30	gfrd-pnc	18	scsd1	10
adlittle	12	greenbea	38	scsd6	11
afiro	9	greenbeb	33	scsd8	9
agg	22	grow15	12	sctap1	13
agg2	18	grow22	12	sctap2	15
agg3	21	grow7	10	sctap3	15
bandm	16	israel	23	share1b	22
beaconfd	8	kb2	17	share2b	14
blend	11	lotfi	14	shell	16
bnl1	25	maros	27	ship04l	13
bnl2	28	modszk1	23	ship04s	17
bore3d	16	perold	42	ship08l	14
brandy	19	pilot.ja	46	ship08s	14
capri	19	pilot.we	48	ship12l	19
cycle	25	pilot4	35	ship12s	17
czprob	32	pilotnov	19	sierra	16
d2q06c	28	recipe	9	stair	16
d6cube	22	sc105	10	standata	9
degen2	13	sc205	11	standgub	9
degen3	19	sc50a	10	standmps	13
e226	18	sc50b	9	stocfor1	10
etamacro	24	scagr25	14	stocfor2	16
fffff800	30	scagr7	13	tuff	21
finnis	19	scfxm1	18	vtp.base	10
fit1d	14	scfxm2	20	wood1p	13
fit1p	13	scfxm3	20	woodw	21
fit2d	18	scorpion	13		

Table 4. Also, similar to Figure 1, problems with infinite condition number are shown in the figure on the far right as a visual aid.

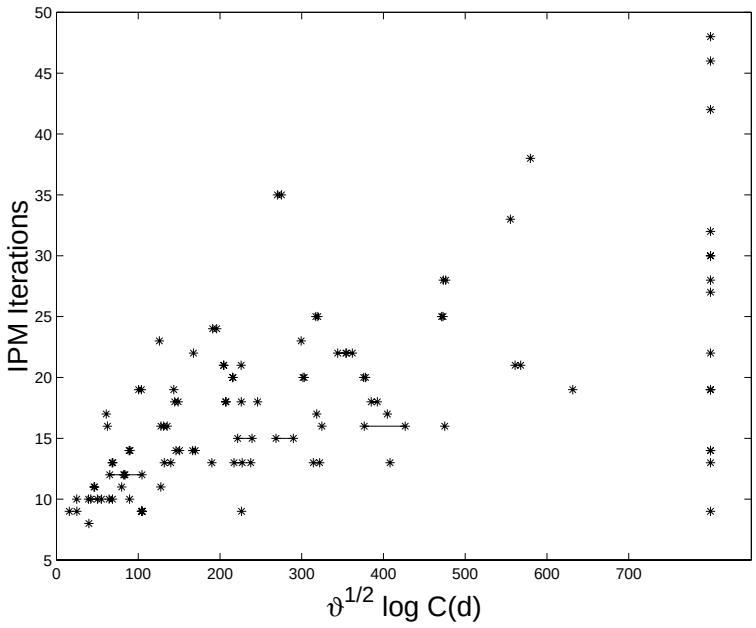

Figure 2: Scatter plot of IPM iterations and $\sqrt{\vartheta}\log C(d)$ for 83 NETLIB problems after pre-processing, using CPLEX 7.1

Figure 2 shows that as $\sqrt{\vartheta}\log C(d)$ increases, so does the number of IPM iterations needed to solve the problem (with exceptions, of course). Perhaps a more accurate summary of the figure is that if the number of IPM iterations is large, then the problem will tend to have a large value of $\sqrt{\vartheta}\log C(d)$. The converse of this statement is not supported by the scatter plot: if a problem has a large value of $\sqrt{\vartheta}\log C(d)$, one cannot state in general that the problem will take a large number of IPM iterations to solve.

In order to be a bit more definitive, we ran a simple linear regression with the IPM iterations of the post-processed problem as the dependent variable and $\sqrt{\vartheta}\log C(d)$ as the independent variable, for the 67 NETLIB problems which have a finite condition number after pre-processing. For the purposes of the regression computation we used the geometric mean of the lower and upper estimates of the condition number from the last two columns of Table 4. The resulting linear regression equation is:

$$\text{IPM Iterations } = 10.8195 + 0.0265\sqrt{\vartheta}\log C(d) \ ,$$

with $R^2 = 0.4267$. This indicates that over 42% of the variation in IPM iteration counts among the NETLIB suite problems is accounted for by $\sqrt{\vartheta}\log C(d)$. A plot of this regression line is shown in Figure 3, where once again the 16 problems that were ill-conditioned are shown in the figure on the far right as a visual aid. Both coefficients of

this simple linear regression are significant at the 95% confidence level, see the regression statistics shown in Table 7.

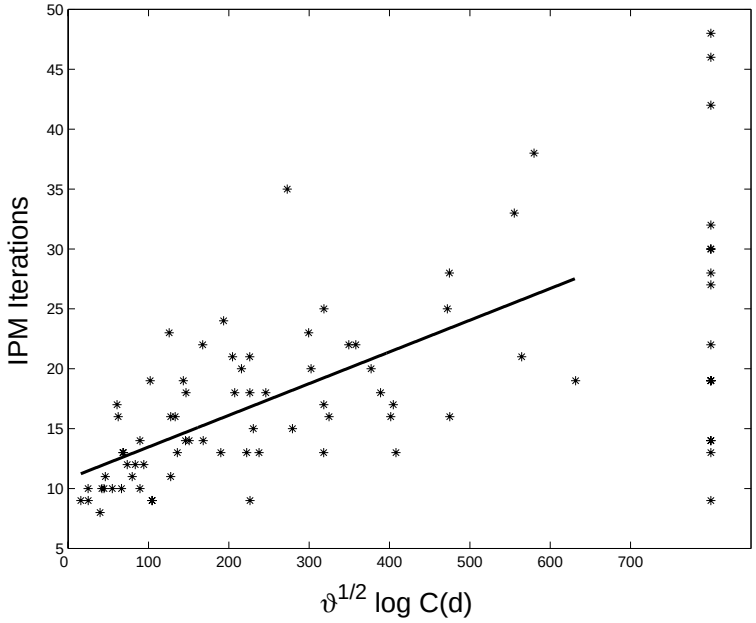

Figure 3: Linear regression of IPM iterations and $\sqrt{\vartheta}\log C(d)$ for 67 NETLIB problems with finite condition number after pre-processing, using CPLEX 7.1 (using the geometric mean of the lower and upper bound estimates of $C(d)$)

Table 7: Statistics for the Linear Regression of IPM iterations and $\sqrt{\vartheta}\log C(d)$.

Coefficient	Value	t-statistic	95% Confidence Interval
β_0	10.8195	10.7044	[8.8009 , 12.8381]
β_1	0.0265	6.9556	[0.0189 , 0.0341]

The presence of $\sqrt{\vartheta}$ in complexity bounds for interior-point methods seems to be a fixture of the theory of self-concordant barrier functions, see [12], despite the belief that such dependence is not borne out in practice. The above regression analysis indicates that $\sqrt{\vartheta}\log C(d)$ does explain 42% of variation in IPM iteration counts among the NETLIB suite of linear optimization problems. Nevertheless, one can also ask whether the condition number alone (without the $\sqrt{\vartheta}$ factor) can account for the variation in IPM iteration counts among the NETLIB suite problems? Figure 4 shows a scatter plot of the number of IPM iterations taken by CPLEX 7.1 to solve the 83 problems in the NETLIB suite after pre-processing and $\log C(d)$ of the post-processed problems (the horizontal lines refer to the range of the lower and upper estimates of $C(d)$ from the

Table 8: Statistics for the Linear Regression of IPM iterations and $\log C(d)$.

Coefficient	Value	t-statistic	95% Confidence Interval
β_0	4.1389	2.1999	[0.3814 , 7.8963]
β_1	1.7591	6.9427	[1.2531 , 2.2652]

last two columns of Table 4; also, problems with infinite condition number are shown in the figure on the far right as a visual aid). We also ran a simple linear regression of IPM iterations as the dependent variable and $\log C(d)$ as the independent variable. The resulting linear regression equation is:

$$\text{IPM Iterations } = 4.1389 + 1.7591 \log C(d) \ ,$$

with $R^2 = 0.4258$. A plot of this regression is shown in Figure 5, and Table 8 shows the regression statistics. It turns out that this regression model is comparable to the linear regression with $\sqrt{\vartheta} \log C(d)$. Both regression models are significant at the 95% confidence level and account for just over 42% of the variance in the iterations of the NETLIB suite. These results indicate that $\log C(d)$ and $\sqrt{\vartheta} \log C(d)$ are essentially equally good at explaining the variation in IPM iteration counts among the NETLIB suite of linear optimization instances.

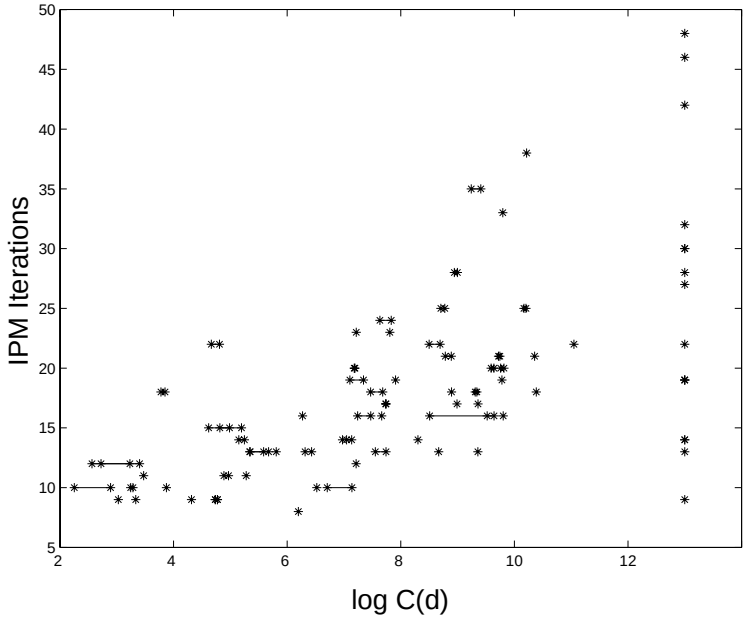

Figure 4: Scatter plot of IPM iterations and $\log C(d)$ for 83 NETLIB problems after pre-processing, using CPLEX 7.1

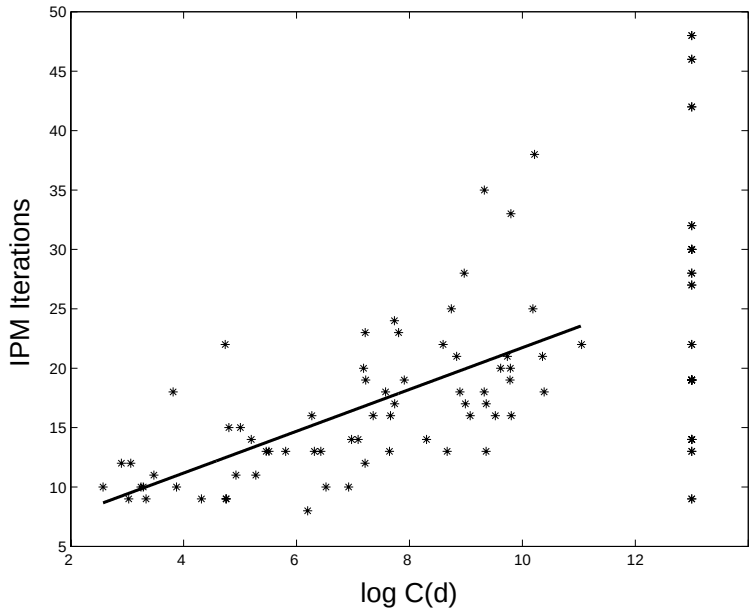

Figure 5: Linear regression of IPM iterations and $\log C(d)$ for 67 NETLIB problems with finite condition number after pre-processing, using CPLEX 7.1 (using the geometric mean of the lower and upper bound estimates of $C(d)$)

We also computed the sample correlation coefficients of the IPM iterations from Table 6 with the following dimensional measures for the 67 finitely-conditioned problems in the NETLIB suite: $\log m$, $\log n$, $\log \vartheta$, and $\sqrt{\vartheta}$. The resulting sample correlations are shown in Table 9. Observe from Table 9 that IPM iterations are better correlated with $\log C(d)$ than with any of the other measures. The closest other measure is $\log m$, for which $R = 0.520$, and so a linear regression of IPM iterations as a function of $\log m$ would yield $R^2 = (0.520)^2 = 0.270$, which is decidedly less than $R^2 = 0.4258$ for $\log C(d)$.

Note from Table 9 that $\log C(d)$ and $\log m$ themselves are somewhat correlated, having a correlation coefficient of 0.431. We have no immediate explanation for this observed correlation, and this may be the subject of future study. Also, note from Table 9 that both $\log \vartheta$ and $\sqrt{\vartheta}$ by themselves are significantly less correlated with the IPM iterations than $\log C(d)$.

4.4 Controlled Perturbations of Problems in the NETLIB Suite

One potential drawback of the analysis in Subsection 4.3 is that in making comparisons of problem instances with different condition numbers one necessarily fails to keep the problem instance size or structure invariant. Herein, we attempt to circumvent this

Table 9: Sample correlations for 67 NETLIB problems (using the geometric mean of the lower and upper bound estimates of $C(d)$)

	IPM iterations	$\log C(d)$	$\log n$	$\log m$	$\log \vartheta$	$\sqrt{\vartheta}$
IPM iterations	1.000					
$\log C(d)$	0.653	1.000				
$\log n$	0.453	0.262	1.000			
$\log m$	0.520	0.431	0.732	1.000		
$\log \vartheta$	0.467	0.267	0.989	0.770	1.000	
$\sqrt{\vartheta}$	0.421	0.158	0.919	0.585	0.929	1.000

drawback by performing controlled perturbations of linear optimization problems which allows one to keep the problem size and structure intact.

Consider a problem instance $d = (A, b, c)$ and the computation of the primal and dual distances to ill-posedness $\rho_P(d)$ and $\rho_D(d)$. It is fairly straightforward to show that if $(i^*, j^*, \lambda^*, (s^+)^*, (s^-)^*, v^*)$ is an optimal solution of (21), then the rank-1 data perturbation:

$$\Delta d = (\Delta A, \Delta b, \Delta c) := \left(-j^* e^{i^*} \left(A^t \lambda^* + (s^+)^* - (s^-)^* \right)^t, -j^* e^{i^*} \left(b^t \lambda^* - v^* \right), 0 \right) \quad (24)$$

is a minimum-norm perturbation for which $\rho_P(d + \Delta d) = 0$ (where e^{i^*} denotes the $(i^*)^{\text{th}}$ unit vector in $\mathbb{R}^m$). That is, $\|\Delta d\| = \rho_P(d)$ and the data instance $\tilde{d} := d + \Delta d$ is primal ill-posed.

The simple construction shown in (24) allows one to construct a controlled perturbation of the data instance d. Consider the family of data instances $d_\alpha := d + \alpha \Delta d$ for $\alpha \in [0, 1]$. Then if $\rho_D(d) \geq \rho_P(d) > 0$ it follows that $\rho(d_\alpha) = (1 - \alpha)\rho(d)$ for $\alpha \in [0, 1]$, and we can bound the condition number of d_α as follows:

$$C(d_\alpha) = \frac{\|d + \alpha \Delta d\|}{(1 - \alpha)\rho(d)} \geq \frac{\|d\| - \alpha \rho(d)}{(1 - \alpha)\rho(d)} ,$$

where the numerator satisfies $\|d\| - \alpha\rho(d) \geq 0$ for $\alpha \in [0, 1]$. In the case when $\|d\| > \rho(d)$ (satisfied by all problem instances in the NETLIB suite) we can create a family of data instances for which $C(d_\alpha) \to \infty$ as $\alpha \to 1$ by varying α in the range $[0, 1]$, all the while keeping the problem dimensions, the structure of the cone C_Y, and the ground-set P invariant.

To illustrate, consider the problem scagr25 from the NETLIB suite, and let $\bar{d}$ denote the data for this problem instance after pre-processing. According to Table 4, $\rho_D(\bar{d}) =$

$0.049075 \geq 0.021191 = \rho_P(\bar{d}) > 0$. Now let $\Delta\bar{d}$ be the perturbation of this data instance according to (24). If we solve the resulting perturbed problem instances $\bar{d}_\alpha$ for select values of $\alpha \in [0, 1]$ and record the number of IPM iterations, we obtain the results portrayed in Figure 6. As the figure shows, the number of IPM iterations grows as the perturbed problem instance becomes more ill-conditioned.

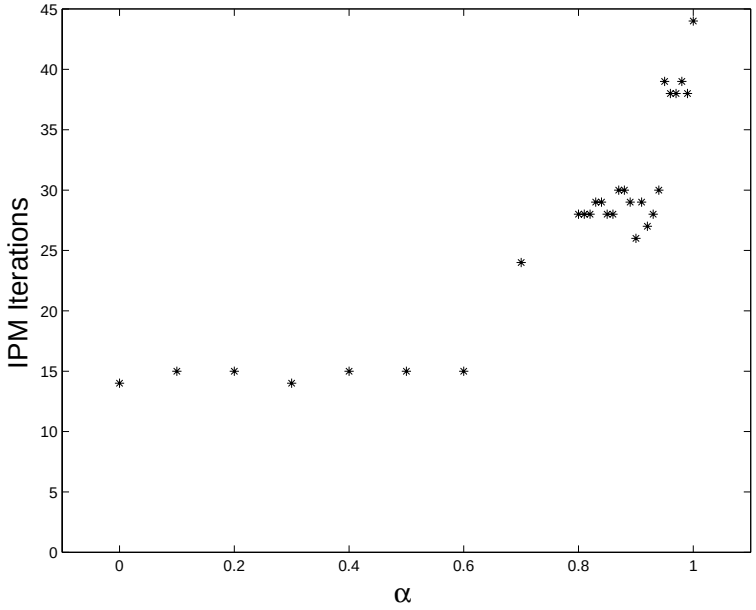

Figure 6: The number of IPM iterations needed to solve the perturbed post-processed problem instance scagr25, as a function of the perturbation scalar α.

The pattern of growth in IPM iterations as the perturbed problem becomes more ill-conditioned is not shared by all problem instances in the NETLIB suite. Figure 7 shows the plot of IPM iterations for problem e226, as the perturbed problem instance becomes more ill-conditioned. For this problem instance the growth in IPM iterations is not monotone.

Of the 67 post-processed problems in the NETLIB suite with finite condition number, 56 of these problems satisfy $\rho_D(d) \geq \rho_P(d) > 0$ and $\|d\| > \rho(d)$, and so are amenable to analysis via the construction described above. For a given problem instance in the NETLIB suite, let k_α denote the number of IPM iterations needed to solve the perturbed post-processed problem instance $\bar{d}_\alpha$. Then

$$\Delta k := k_1 - k_0$$

is the difference between the IPM iterations needed to solve the un-perturbed problem instance and the fully-perturbed problem instance. Table 10 shows some summary statistics of the distribution of Δk for the 56 problems in the NETLIB suite that are readily

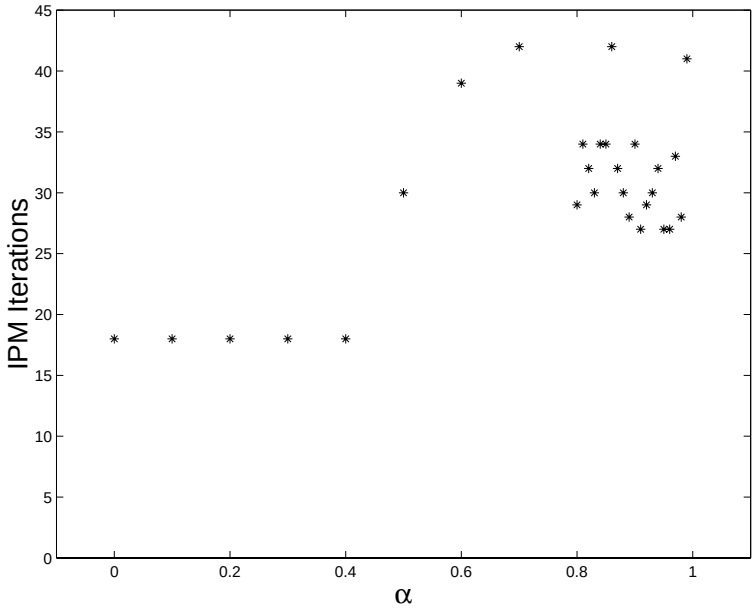

Figure 7: The number of IPM iterations needed to solve the perturbed post-processed problem instance e226, as a function of the perturbation scalar α.

amenable to this analysis. As the table shows, the fully perturbed problem instance has a larger IPM iteration count in 68% (38 out of 56) of the problem instances. Curiously, the number of IPM iterations is actually less (but by at most three iterations) for the fully-perturbed problem instance in 18% (10 out of 56) problem instances amenable to this analysis. A rough summary of the results in Table 10 is that the number of IPM iterations for the fully perturbed problem increases dramatically (more than 10 iterations) on 30% of the problem instances, increases modestly (1-10 iterations) on 38% of the problem instances, and remains the same or decreases slightly on 32% of problem instances.

5 Summary Conclusions

The purpose of this paper has been to gain some computational experience and to test the practical relevance of condition numbers for linear optimization on problem instances that one might encounter in practice. We used the NETLIB suite of linear optimization problems as a test bed for condition number computation and analysis, and we computed condition numbers for 83 NETLIB suite problem instances both prior to and after the instance was pre-processed using CPLEX 7.1. This computation was done using the ground-set model format of convex optimization, where the ground-set was defined by

Table 10: The distribution of the change in IPM iterations needed to solve the unperturbed problem instance and the fully-perturbed problem instance, for 56 post-processed problems in the NETLIB suite.

Change in IPM Iterations (Δk)	Number of Problem Instances
-3 to -1	10
0	8
1 to 5	12
6 to 10	9
11 or more	17
Total	56

the lower and upper bound constraints on the variables.

A summary of our computational findings is as follows:

1. 72% of the original problem instances in the NETLIB suite are ill-conditioned.

2. 70% of the original problem instances in the NETLIB suite are primal ill-posed, i.e., arbitrarily small data perturbations will render the primal problem infeasible.

3. After pre-processing of the problem instances by CPLEX 7.1, only 19% of problem instances are ill-posed.

4. $\log C(d)$ of the 67 post-processed problems with finite condition number is fairly nicely distributed in the range from $2.6 - 11.0$.

5. The number of IPM iterations needed to solve linear optimization problem instances is related to the condition numbers of the post-processed problem instances. If the number of IPM iterations is large for a given problem instance, then the problem will tend to have a large post-processed condition number. However, the converse of this statement is not supported by computational experience: if the post-processed problem instance has a large condition number, one cannot assert that the problem instance will need a large number of IPM iterations to solve.

6. A simple linear regression model of IPM iterations as the dependent variable and $\sqrt{\vartheta}\log C(d)$ as the independent variable yields a positive linear relationship between IPM iterations and $\sqrt{\vartheta}\log C(d)$, significant at the 95% confidence level, with $R^2 = 0.4267$. This means that 42% of the variation in IPM iterations among the NETLIB suite problems is accounted for by $\sqrt{\vartheta}\log C(d)$.

7. A simple linear regression model of IPM iterations as the dependent variable and $\log C(d)$ as the independent variable yields a very similar result, also significant at the 95% confidence level, and with $R^2 = 0.4258$. These results indicate that $\log C(d)$ and $\sqrt{\vartheta} \log C(d)$ are essentially equally good at explaining the variation in IPM iteration counts among the NETLIB suite of linear optimization instances.

8. The number of IPM iterations correlates better with $\log C(d)$ than with $\log n$, $\log m$, $\log \vartheta$, or $\sqrt{\vartheta}$.

9. Curiously, $\log C(d)$ is somewhat correlated with $\log m$, having a sample correlation of 0.431. This observation bears further scrutiny.

10. In controlled perturbations of problem instances to ill-conditioned perturbed instances, the number of IPM iterations of the ill-posed perturbed instances are larger than for the original instance in about 68% of the problems studied, significantly larger in about half of these. However in the other 32% of the problems studied there was no change or even a slight decrease in IPM iterations.

References

[1] F. Cucker and J. Peña. A primal-dual algorithm for solving polyhedral conic systems with a finite-precision machine. Technical report, GSIA, Carnegie Mellon University, 2001.

[2] M. Epelman and R. M. Freund. A new condition measure, preconditioners, and relations between different measures of conditioning for conic linear systems. *SIAM Journal on Optimization*, 12(3):627–655, 2002.

[3] S. Filipowski. On the complexity of solving sparse symmetric linear programs specified with approximate data. *Mathematics of Operations Research*, 22(4):769–792, 1997.

[4] S. Filipowski. On the complexity of solving feasible linear programs specified with approximate data. *SIAM Journal on Optimization*, 9(4):1010–1040, 1999.

[5] R. Freund and M. Todd. Barrier functions and interior–point algorithms for linear programming with zero–, one–, or two–sided bounds on the variables. *Mathematics of Operations Research*, 20(2):415–440, 1995.

[6] R. M. Freund and J. R. Vera. Condition-based complexity of convex optimization in conic linear form via the ellipsoid algorithm. *SIAM Journal on Optimization*, 10(1):155–176, 1999.

[7] R. M. Freund and J. R. Vera. On the complexity of computing estimates of condition measures of a conic linear system. Technical Report, Operations Research Center, MIT, August 1999.

[8] R. M. Freund and J. R. Vera. Some characterizations and properties of the "distance to ill-posedness" and the condition measure of a conic linear system. *Mathematical Programming*, 86(2):225–260, 1999.

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

[10] I. Lustig, R. Marsten, and D. Shanno. The primal dual interior point method on the cray supercomputer. In T. F. Coleman and Y. Li, editors, *Large-Scale Numerical Optimization, Papers from the Workshop held at Cornell University, Ithaca, NY, October 1989*, volume 46 of *SIAM Proceedings in Applied Mathematics*, pages 70–80. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1990.

[11] I. Lustig, R. Marsten, and D. Shanno. Interior point methods: computational state of the art. *ORSA Journal on Computing*, 6:1–14, 1994.

[12] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1993.

[13] NETLIB linear programming library. http://www.netlib.org/lp/.

[14] M. A. Nunez and R. M. Freund. Condition measures and properties of the central trajectory of a linear program. *Mathematical Programming*, 83(1):1–28, 1998.

[15] M. A. Nunez and R. M. Freund. Condition-measure bounds on the behavior of the central trajectory of a semi-definite program. *SIAM Journal on Optimization*, 11(3):818–836, 2001.

[16] F. Ordóñez. *On the Explanatory Value of Condition Numbers for Convex Optimization: Theoretical Issues and Computational Experience*. PhD thesis, Massachusetts Institute of Technology, 2002. In preparation.

[17] J. Peña. Computing the distance to infeasibility: theoretical and practical issues. Technical report, Center for Applied Mathematics, Cornell University, 1998.

[18] J. Peña and J. Renegar. Computing approximate solutions for convex conic systems of constraints. *Mathematical Programming*, 87(3):351–383, 2000.

[19] J. Renegar. A polynomial-time algorithm, based on Newnton's method, for linear programming. *Mathematical Programming*, 40(1):59–93, 1988.

[20] J. Renegar. Some perturbation theory for linear programming. *Mathematical Programming*, 65(1):73–91, 1994.

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

[22] R. Shamir. The efficiency of the simplex method: a survey. *Management Science*, 33(3):301–334, 1987.

[23] J. R. Vera. Ill-posedness and the computation of solutions to linear programs with approximate data. Technical Report, Cornell University, May 1992.

[24] J. R. Vera. *Ill-Posedness in Mathematical Programming and Problem Solving with Approximate Data*. PhD thesis, Cornell University, 1992.

[25] J. R. Vera. Ill-posedness and the complexity of deciding existence of solutions to linear programs. *SIAM Journal on Optimization*, 6(3):549–569, 1996.

[26] J. R. Vera. On the complexity of linear programming under finite precision arithmetic. *Mathematical Programming*, 80(1):91–123, 1998.