SIAM J. OPTIM.
Vol. 14, No. 2, pp. 307–333

COMPUTATIONAL EXPERIENCE AND THE EXPLANATORY VALUE OF CONDITION MEASURES FOR LINEAR OPTIMIZATION*

FERNANDO ORDÓÑEZ† AND ROBERT M. FREUND‡

Abstract. The modern theory of condition measures for convex optimization problems was initially developed for convex problems in the conic format

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

and several aspects of the theory have now been extended to handle nonconic formats as well. In this theory, the (Renegar) condition measure $C(d)$ for (CP_d) has been shown to be connected to bounds on 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). Herein we test the practical relevance of the condition measure theory, as applied to linear optimization problems that one might typically encounter in practice. Using the NETLIB suite of linear optimization problems as a test bed, we found that 71% of the NETLIB suite problem instances have infinite condition measure. In order to examine condition measures of the problems that are the actual input to a modern interior-point-method (IPM) solver, we also computed condition measures for the NETLIB suite problems after preprocessing by CPLEX 7.1. Here we found that 19% of the postprocessed problem instances in the NETLIB suite have infinite condition measure, and that $\log C(d)$ of the postprocessed problems is fairly nicely distributed. Furthermore, among those problem instances with finite condition measure after preprocessing, there is a positive linear relationship between IPM iterations and $\log C(d)$ of the postprocessed problem instances (significant at the 95% confidence level), and 42% of the variation in IPM iterations among these NETLIB suite problem instances is accounted for by $\log C(d)$ of the postprocessed problem instances.

Key words. condition measure, interior-point method, linear programming, computation, preprocessing

AMS subject classifications. 90-04, 90C05, 90C60, 90C51

DOI. 10.1137/S1052623402401804

1. Introduction.

The modern theory of condition measures for convex optimization problems was initially developed in [24] for problems in the following conic format:

$$(\mathrm{CP}_d) \qquad \begin{array}{ll} z_* := \min & c^t x \\ \text{s.t.} & Ax - b \in C_Y, \\ & x \in C_X, \end{array} \tag{1.1}$$

where, for concreteness, we consider A to be an $m \times n$ real matrix; $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$; $C_X \subseteq \mathbb{R}^n$, $C_Y \subseteq \mathbb{R}^m$ 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 $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively, and let $\|A\|$ denote the usual operator norm; let $\|v\|_*$ denote the dual norm associated with the norm $\|w\|$ on $\mathbb{R}^n$ or $\mathbb{R}^m$. We define the norm of the data instance $d = (A, b, c)$ by $\|d\| := \max\{\|A\|, \|b\|, \|c\|_*\}$.

The theory of condition measures for (CP_d) focuses on three measures, $\rho_P(d)$, $\rho_D(d)$, and $C(d)$, to bound various behavioral and computational quantities pertaining

*Received by the editors January 31, 2002; accepted for publication (in revised form) February 10, 2003; published electronically August 22, 2003. This research was partially supported through the Singapore-MIT Alliance.

http://www.siam.org/journals/siopt/14-2/40180.html

†Industrial and Systems Engineering, University of Southern California, GER-247, Los Angeles, CA 90089-0193 (fordon@usc.edu).

‡MIT Sloan School of Management, 50 Memorial Drive, Cambridge, MA 02142 (rfreund@mit.edu).

to (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):

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

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

$$(\mathrm{CD}_d) \qquad \begin{array}{rl} z^* := \max & b^t y \\ \text{s.t.} & c - A^t y \in C_X^*, \\ & y \in C_Y^*, \end{array} \tag{1.2}$$

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

$$C(d) := \frac{\|d\|}{\min\{\rho_P(d), \rho_D(d)\}}; \tag{1.3}$$

a problem is called "ill-posed" if $\min\{\rho_P(d), \rho_D(d)\} = 0$, equivalently, $C(d) = \infty$. These three condition measure quantities have been shown in theory to be connected to a wide variety of bounds on 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 [24], [5], [4], [8], [9], [10], [29], [27], [30], [28], [20], [22]. In the context of interior-point methods for linear and semidefinite optimization, these same three condition measures have also been shown to be connected to various quantities of interest regarding the central trajectory; see [16] and [17]. The connection of these condition measures to the complexity of algorithms has been shown in [8], [9], [25], [2], [3], and some of the references contained therein. While this literature has focused almost exclusively on the conic format of (1.1), there have been some attempts to extend the theory to convex problems in structured nonconic formats; see Filipowski [4], Peña [21] and [19], and [18].

Given the theoretical importance of these many results, it is natural to ask what typical values of these condition measures might arise in practice. Are such problems typically well-posed or ill-posed? How are the condition measures of such problems distributed? We begin to answer these questions in this paper, where we compute and analyze these three condition measures for the NETLIB suite of industrial and academic linear programming (LP) problems. We present computational results that indicate that 71% of the NETLIB suite of linear optimization problem instances are ill-posed, i.e., have infinite condition measure; see section 4.1.

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, over a huge range of problem sizes; see [13], for example. Using the condition-measure 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 [25] 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 to what extent the observed variation in the number of IPM iterations (already small) can be accounted for by the condition measures of the LP problems that are solved. In order to answer this question, first note that typical IPM solvers perform routine preprocessing to modify the LP problem prior to solving. In order to examine condition measures of the problems that are the actual input to a modern IPM solver, we computed condition measures for the NETLIB suite problems after prepreprocessing by CPLEX 7.1. We found that 19% of the postprocessed problem instances in the NETLIB suite have infinite condition measure, and that $\log C(d)$ of the postprocessed problems is fairly nicely distributed; see section 4.2. In section 4.3, we show that, among the 72 postprocessed problem instances in the NETLIB suite with finite condition measure, the number of IPM iterations needed to solve these problems varies roughly linearly (and monotonically) with $\log C(d)$ of the postprocessed 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 postprocessed problem instances, significant at the 95% confidence level, with $R^2 = 0.4160$. Therefore, in the sample of 72 NETLIB suite problem instances whose postprocessed condition measure is finite, about 42% of the variation in IPM iterations among these problems is accounted for by $\log C(d)$ of the problem instances after preprocessing. Additionally, $\log C(d)$ correlates with IPM iterations better than any other problem measure; see section 4.3.

The organization of this paper is as follows. In section 2, we lay the groundwork for the computation of condition measures for the NETLIB suite. Section 3 describes our methodology for computing condition measures, and section 4 contains the computational results. Section 5 contains some discussion and open questions.

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 measures, one can start by computing the condition measures 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 [15]. Practical methods for computing (or approximately computing) condition measures for convex optimization problems in conic format (CP_d) have been developed in [9] and [20], and such methods are relatively easy to implement. It would then seem to be a simple task to compute condition measures 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 measure theory just a bit, as we now explain.

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

$$\begin{array}{lll} \min\limits_{x} & c^t x & \\ \text{s.t.} & A_i x \le b_i, & i \in L, \\ & A_i x = b_i, & i \in E, \\ & A_i x \ge b_i, & i \in G, \\ & x_j \ge l_j, & j \in L_B, \\ & x_j \le u_j, & j \in U_B, \end{array} \tag{2.1}$$

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 "$b_{i,l} \leq A_i x \leq b_{i,u}$." We ignore this for now.) By defining C_Y to be an appropriate Cartesian product of nonnegative halflines $\mathbb{R}_+$, nonpositive halflines $-\mathbb{R}_+$, and the origin $\{0\}$, we can naturally consider the constraints to be in the conic format "$Ax-b \in C_Y$," where $C_Y \subset \mathbb{R}^m$ 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 measures. One way is to convert the lower and upper bound constraints into ordinary constraints, whose conversion of (2.1) to conic format is

$$\begin{array}{rll} P_1 : \min\limits_{x} & c^t x & \\ \text{s.t.} & Ax - b \in C_Y, & \\ & Ix - l \geq 0, & \\ & Ix - u \leq 0, & \end{array}$$

whose data for this now-conic format is

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

with cones

$$\bar{C}_Y := C_Y \times \mathbb{R}^n_+ \times -\mathbb{R}^n_+ \quad \text{and} \quad \bar{C}_X := \mathbb{R}^n.$$

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

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

whose data for this now-conic format is

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

with cones

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

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

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

TABLE 2.1
Condition measures for two different conic conversions of the same problem, using the L_∞-norm in the space of the variables and the L_1-norm in the space of the right-hand-side vector.

	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 2.1 shows condition measures 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 measures is described in Remark 6 of [10].) As Table 2.1 shows, the choice of conversion strategy can have a very large impact on the resulting condition measures, thereby calling into question the practical significance of performing such conversions to conic format.

2.1. Structured formats for optimization. The analysis presented above indicates a need for extending condition-measure concepts to problems with structured nonconic formats, and indeed there has been some research along this line. Filipowski [4] examines the efficiency of solving symmetric-form linear programs whose sparsity pattern is not subject to modification, and Peña [21] develops condition measures for conic problems where certain rows and columns of data are not subject to modification; the latter can be used directly or indirectly to construct condition measures for many types of structured nonconic problems. More recently, in [18], the theory of condition measures and their properties has been extended from the conic format to handle more general structured convex optimization in the following "ground-set" format:

$$z_*(d) = \begin{array}{ll} \min & c^t x \\ \text{s.t.} & Ax - b \in C_Y, \\ & x \in P, \end{array} \qquad (\mathrm{GP}_d) \tag{2.2}$$

where P is called the ground set; P is no longer required to be a cone, but instead can be any closed convex set. In practical applications, P could be chosen to be the solution of lower and upper bound constraints $l \le x \le u$, or P could be a convex cone C_X, or P could perhaps be the solution to network flow constraints of the form $Nx = b$, $x \ge 0$, etc. The set P (and the cone C_Y) remains fixed as part of the definition of the problem, and the description of P is not part of the data $d = (A, b, c)$. Many aspects of the theory of condition measures for conic convex optimization have been extended to the more general ground-set model format (2.2); see [18]. We will use this ground-set format in our computation and evaluation of condition measures for linear programs that arise in practice.

In treating linear programs (2.1) as instances in the format (2.2), there is some leeway as to what structure to place in the ground set P. One strategy is to define P simply by the lower and upper bounds,

$$P := \{x \mid x_j \ge l_j \text{ for } j \in L_B,\ x_j \le u_j \text{ for } j \in U_B\}, \tag{2.3}$$

and then rewrite the other constraints in conic format as described earlier. In this approach the lower and upper bounds are handled conveniently, although the data d does not then 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 not generally thought of as subject to data modification. Of course, there are other settings where keeping the lower and upper bounds fixed independent of the other constraints is not as natural.

Another strategy would be to try to examine the individual constraints of the LP instance in order to identify specific structures to include in P. For example, in addition to lower and upper bound constraints, a particular LP instance might also have some network constraints, or might have generalized upper bound (GUB) constraints of the form $\sum_{j\in J} x_j \leq M$, variable upper bound constraints $x_j \leq x_k$, etc. Constraints of this type have no inherent data that one would think of as subject to possible modification; therefore they could be included in the set P.

In order to develop some computational experience with condition measures for the NETLIB suite, we chose the more straightforward strategy of defining P only by the upper and lower bounds of the LP instance (2.3). We chose this approach because (2.3) best reflects the types of LP structures that are explicitly treated algorithmically in modern simplex and IPM software, and because we had minimal foreknowledge of any explicit structures of individual linear prgrams in the NETLIB suite.

(In the related area of robust optimization, Ben-Tal and Nemirovski test robust optimization methodologies on the NETLIB suite by attempting to identify individual data entries of linear inequalities (but not equalities) in constraints of NETLIB suite linear programs that might be subject to data modification or data error; see [1].)

2.2. Definition of $C(d)$ for ground-set format. The general set-up for the development of condition-measure theory for the ground-set model format is developed in [18]. We review this material briefly here for completeness.

Let X_d denote the feasible region of (GP_d),

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

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

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

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.

Let R denote the recession cone of P, namely,

$$R := \{v \mid \text{there exists } x \in P \text{ for which } x + \theta v \in P \text{ for all } \theta \geq 0\}. \tag{2.4}$$

Since P is a closed convex set, the recession cone 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

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

The Lagrange dual of (GP_d) is

$$(\mathrm{GD}_d) \qquad \begin{array}{rll} z^*(d) = & \max\limits_{y,v} & b^t y - v \\ & \text{s.t.} & (c - A^t y, v) \in C^*, \\ & & y \in C_Y^*. \end{array} \tag{2.5}$$

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

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

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

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

The condition measures $\rho_P(d), \rho_D(d)$ are shown in [18] 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, aspect ratios of inscribed balls, deformation of the feasible region under perturbation, and the complexity of interior-point algorithms.

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

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

For $d \in \mathcal{F}$, the definition of the condition measure 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 (positive) scale invariant reciprocal of the distance to the set of data instances that are either primal or dual infeasible, and $\rho(d) := \min\{\rho_P(d), \rho_D(d)\}$ 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 [9]. 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 $2n + 2m$ linear programs of size roughly that of the original problem. As is discussed in [9], 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 of $\rho_P(d)$ and $\rho_D(d)$. We begin our analysis with a seemingly innocuous proposition which will prove to be very useful.

PROPOSITION 3.1. *Consider the problem*

$$\begin{array}{rll} z_1 = & \min\limits_{v,w} & f(v,w) \\ & \text{s.t.} & \|v\|_\infty = 1, \\ & & (v,w) \in K, \end{array} \tag{3.1}$$

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 problems (3.1) *and* (3.2) *have the same optimal values, i.e., $z_1 = z_2$, where*

$$\begin{array}{rll} z_2 = & \min\limits_{i \in \{1,\ldots,n\}, j \in \{-1,1\}} & \min\limits_{v,w} f(v,w) \\ & & v_i = j, \\ & & (v,w) \in K. \end{array} \tag{3.2}$$

Proof. Let (v^*, w^*) be an optimal solution of (3.1). 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 (3.2) for $i = i^*$ and $j = j^*$, and so $z_2 \le z_1$.

If (v^*, w^*) is an optimal solution of (3.2) with $i = i^*$ and $j = j^*$, then $\|v^*\|_\infty \ge 1$. If $\|v^*\|_\infty = 1$, the point (v^*, w^*) is feasible for (3.1), 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 (3.2) for some $i = \hat{i} \neq i^*$ and $j = \hat{j}$, and so $z_2 \le f(\tilde{v}, \tilde{w}) = f(\frac{1}{\|v^*\|_\infty}(v^*, w^*)) = \frac{1}{\|v^*\|_\infty} f(v^*, w^*) \le z_2$, which now implies that $(\tilde{v}, \tilde{w})$ is also an optimal solution of (3.2). Since $\|\tilde{v}\|_\infty = 1$, the previous argument implies that $z_1 \le z_2$, completing the proof. ∎

3.1. Computing $\rho_P(d)$ and $\rho_D(d)$. The following theorem, which is proved in [18], characterizes $\rho_P(d)$ and $\rho_D(d)$ as the optimal solution values of certain optimization problems. In this theorem, recall from (2.4) that R denotes the recession cone of the ground set P.

THEOREM 3.2 (Theorems 5 and 6 of [18]). *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*

$$\begin{array}{rll} j_P(d) = & \min\limits_{y,s,v} & \max\{\|A^t y + s\|_1, |b^t y - v|\} \\ & \text{s.t.} & \|y\|_\infty = 1, \\ & & y \in C_Y^*, \\ & & (s,v) \in C^*, \end{array} \tag{3.3}$$

and

$$\begin{array}{rll} j_D(d) = & \min\limits_{x,p,g} & \max\{\|Ax - p\|_1, |c^t x + g|\} \\ & \text{s.t.} & \|x\|_\infty = 1, \\ & & x \in R, \\ & & p \in C_Y, \\ & & g \ge 0. \end{array} \tag{3.4}$$

Neither (3.3) nor (3.4) are convex problems. However, both (3.3) and (3.4) are of the form (3.1), and so we can invoke Proposition 3.1 and solve (3.3) and (3.4) using

problem (3.2). From Proposition 3.1, we have

$$\begin{array}{rlll} \rho_P(d) = & \min\limits_{i\in\{1,\ldots,m\},j\in\{-1,1\}} & \min\limits_{y,s,v} & \max\{\|A^t y + s\|_1, |b^t y - v|\} \\ & & \text{s.t.} & y_i = j, \\ & & & y \in C_Y^*, \\ & & & (s,v) \in C^*, \end{array} \tag{3.5}$$

and

$$\begin{array}{rlll} \rho_D(d) = & \min\limits_{i\in\{1,\ldots,n\},j\in\{-1,1\}} & \min\limits_{x,p,g} & \max\{\|Ax - p\|_1, |c^t x + g|\} \\ & & \text{s.t.} & x_i = j, \\ & & & x \in R, \\ & & & p \in C_Y, \\ & & & g \geq 0. \end{array} \tag{3.6}$$

Taken together, (3.5) and (3.6) 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 roughly the same size as the original problem (GP_d).

Of course, each of the $2n + 2m$ convex problems in (3.5) and (3.6) 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 (2.1) there are convenient linear inequality characterizations of all of the cones involved in (3.5) and (3.6). The cone C_Y is easily seen to be

$$C_Y = \{p \in \mathbb{R}^m \mid p_i \leq 0 \text{ for } i \in L, p_i = 0 \text{ for } i \in E, p_i \geq 0 \text{ for } i \in G\}, \tag{3.7}$$

and so

$$C_Y^* = \{y \in \mathbb{R}^m \mid y_i \leq 0 \text{ for } i \in L, y_i \in \mathbb{R} \text{ for } i \in E, y_i \geq 0 \text{ for } i \in G\}. \tag{3.8}$$

With the ground set P defined in (2.3), we have

$$R = \{x \in \mathbb{R}^n \mid x_j \geq 0 \text{ for } j \in L_B,\ x_j \leq 0 \text{ for } j \in U_B\} \tag{3.9}$$

and also

$$C = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} \mid t \geq 0, x_j \geq l_j t \text{ for } j \in L_B,\ x_j \leq u_j t \text{ for } j \in U_B\}. \tag{3.10}$$

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 $(s, v, s^+, s^-) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n$:

$$\begin{array}{rcll} s - s^+ + s^- & = & 0, & \\ s^+ & \geq & 0, & \\ s^- & \geq & 0, & \\ s_j^- & = & 0 & \text{for } j \in N \setminus U_B, \\ s_j^+ & = & 0 & \text{for } j \in N \setminus L_B, \\ v + \sum\limits_{j\in L_B} l_j s_j^+ - \sum\limits_{j\in U_B} u_j s_j^- & \geq & 0, & \end{array} \tag{3.11}$$

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 3.3. *For the ground set P defined in* (2.3)*, the cone C^* is characterized by*

$$C^* = \{(s, v) \in \mathbb{R}^n \times \mathbb{R} \mid (s, v, s^+, s^-) \text{ satisfies (3.11) for some } s^+, s^- \in \mathbb{R}^n\}.$$

Proof. Suppose first that (s, v) together with some s^+, s^- satisfies (3.11). Then for all $(x, t) \in C$ we have

$$\begin{aligned} (x,t)^t(s,v) &= \sum_{j \in L_B} s_j^+ x_j - \sum_{j \in U_B} s_j^- x_j + tv \\ &\geq \sum_{j \in L_B} s_j^+ l_j t - \sum_{j \in U_B} s_j^- u_j t + tv \qquad (3.12) \\ &\geq 0, \end{aligned}$$

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

$$\begin{aligned} -\infty < -v \leq \min_{x \in P} s^t x = \min \quad & \sum_{j=1}^{n} s_j x_j \\ \text{s.t.} \quad & x_j \geq l_j \text{ for } j \in L_B, \qquad (3.13) \\ & x_j \leq u_j \text{ for } j \in U_B, \end{aligned}$$

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 (3.13) 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 (3.11), whereby (s, v, s^+, s^-) satisfies all inequalities of (3.11). ∎

Taken together, we can use (3.7), (3.8), (3.9), (3.10), and Proposition 3.3 to rewrite the right-most minimization problems of (3.5) and (3.6) and obtain

$$\begin{aligned} \rho_P(d) = \min_{\substack{i \in \{1,\ldots,m\} \\ j \in \{-1,1\}}} \min_{y, s^+, s^-, v} \quad & \max\{\|A^t y + s^+ - s^-\|_1, |b^t y - v|\} \\ \text{s.t.} \quad & y_i = j, \\ & y_l \leq 0 \quad \text{for } l \in L, \\ & y_l \geq 0 \quad \text{for } l \in G, \\ & s_k^- = 0 \quad \text{for } k \in N \setminus U_B, \qquad (3.14) \\ & s_k^+ = 0 \quad \text{for } k \in N \setminus L_B, \\ & v + \sum_{k \in L_B} l_k s_k^+ - \sum_{k \in U_B} u_k s_k^- \geq 0, \\ & s^+, s^- \geq 0, \end{aligned}$$

and

$$\begin{aligned} \rho_D(d) = \min_{\substack{i \in \{1,\ldots,n\} \\ j \in \{-1,1\}}} \min_{x, p, g} \quad & \max\{\|Ax - p\|_1, |c^t x + g|\} \\ \text{s.t.} \quad & x_i = j, \\ & x_k \geq 0 \quad \text{if } k \in L_B, \\ & x_k \leq 0 \quad \text{for } k \in U_B, \qquad (3.15) \\ & p_l \leq 0 \quad \text{for } l \in L, \\ & p_l = 0 \quad \text{for } l \in E, \\ & p_l \geq 0 \quad \text{for } l \in G, \\ & g \geq 0, \end{aligned}$$

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 $2n+2m$ linear programs, under the choice of norms given in Assumption 1.

3.2. Computing $\|d\|$. In order to compute the condition measure $C(d) := \|d\|/\rho(d)$, 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 NP-hard (one can easily pose MAXCUT as a special case). We therefore will bound $\|A\|_{\infty,1}$ and hence $\|d\|$ from below and above, using the following elementary norm inequalities:

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

where

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

$e := (1, \ldots, 1)^t$, 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 measures for the NETLIB suite prior to preprocessing. We chose the NETLIB suite of linear optimization problem instances as a representative suite of LP problems encountered in practice, and we computed the condition measures $\rho_P(d), \rho_D(d)$, and $C(d)$ for problem instances in this suite using the methodology developed in section 3. The NETLIB suite is comprised of 98 linear optimization problems 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, and the sizes and diversity of the problems contained therein seem to be representative of general practice. The sizes of the problem instances in the NETLIB suite range from 32 variables and 28 constraints to problems with roughly 9,000 variables and 3,000 constraints. 44 of the 98 problems in the suite have nonzero 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 (range constraints do not naturally fit into either the conic model or the ground-set model format). On four of the remaining problems (dfl001, qap12, qap15, stocfor3) our methodology has not yet exhibited convergence to a solution, and these four problems were omitted as well, yielding a final sample set of 89 linear optimization problems. The burden of computing the distances to ill-posedness for the NETLIB suite via the solution of $2n + 2m$ linear programs 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 `maros-r7` ($n = 9,408$ and $m = 3,136$), the total computation time was $240,627.59$ seconds of machine time (66.84 hours).

Table 4.1 shows the distances to ill-posedness and the condition measure estimates for the 89 problems, using the methodology for computing $\rho_P(d)$ and $\rho_D(d)$ and for estimating $\|d\|$ presented in section 3. All LP computation was performed using CPLEX 7.1 (function *primopt*).

Table 4.2 presents some summary statistics of the condition measure computations from Table 4.1. As the table shows, 71% (63/89) of the problems in the NETLIB suite are ill-posed due to either $\rho_P(d) = 0$ or $\rho_D(d) = 0$ or both. Furthermore, notice that, among these 63 ill-posed problems, almost all (61 out of 63) have $\rho_P(d) = 0$. This means that for 69% (61/89) of the problems in the NETLIB suite, arbitrarily small changes in the data will render the primal problem infeasible.

Notice from Table 4.1 that there are three problems for which $\rho_D(d) = \infty$, namely, `fit1d`, `fit2d`, and `sierra`. This can happen only when the ground set P is bounded, which for linear optimization means that all variables have finite lower and upper bounds.

The computational results in Tables 4.1 and 4.2 have shown that 61 of the 89 linear programs in the NETLIB suite are primal ill-posed, i.e., $\rho_P(d) = 0$, and so arbitrarily small changes in the data will render the primal problem infeasible. For feasible linear programs, $\rho_P(d) = 0$ can happen only if (i) there are linear dependencies among the equations of the problem instance (2.1), or (ii) there is an implied reverse inequality among the inequalities and lower and upper bounds of the problem instance. Furthermore, it is easy to show that if $s = 0$ in an optimal solution of (3.3), then there are linear dependencies in the equations (and possibly implied reverse inequalities as well), whereas if $s \neq 0$ in an optimal solution of (3.3), then there is an implied reverse inequality (and possibly linear dependencies as well). This then can be used to evaluate the causes of the ill-posedness of the 61 primal ill-posed instances. We examined the optimal solutions of (3.3) for the 61 primal ill-posed linear programs in the NETLIB suite in order to evaluate the causes of the ill-posedness among these problems. Table 4.3 summarizes our findings, which show that for at least 34% of the primal ill-posed problem instances there are linear dependencies among the equations of (2.1).

4.2. Condition measures for the NETLIB suite after preprocessing. Most commercial software packages for solving linear optimization problems perform preprocessing heuristics prior to solving a problem instance. These heuristics typically include checks for eliminating linearly dependent equations, heuristics for identifying and eliminating redundant variable lower and upper bounds, and rules for row and/or column rescaling, etc. The purposes of the preprocessing are to reduce the size of the problem instance by eliminating dependent equations and redundant inequalities, and to improve numerical computation and enhance iteration performance by rescaling of rows and/or columns. The original problem instance is converted to a postprocessed instance by the processing heuristics, and it is this postprocessed instance that is used as input to solution software. In CPLEX 7.1, the postprocessed problem can be accessed using function *prslvwrite*. This function writes the postprocessed problem to disk, whence it can be read.

In order to examine condition measures of the problems that are the actual input to a modern IPM solver, we computed condition measures for the NETLIB suite problems after preprocessing by CPLEX 7.1. The processing used was the default CPLEX preprocessing with the linear dependency check option activated. Table 4.4 shows the condition measures in detail for the postprocessed versions of the problems, and Table 4.5 presents some summary statistics of these condition measures. Notice

TABLE 4.1

Condition measures for the NETLIB suite LP problem instances (prior to preprocessing by CPLEX 7.1).

			$\|d\|$		$\log C(d)$	
Problem	$\rho_P(d)$	$\rho_D(d)$	Lower bound	Upper bound	Lower bound	Upper bound
25fv47	0.000000	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
fit2p	1.057333	1.000000	621,470	658,700	5.8	5.8
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	∞	∞
maros	0.000000	0.000000	2.51E+06	2.55E+06	∞	∞
maros-r7	1.000000	0.628096	1.02E+07	1.02E+07	7.2	7.2
modszk1	0.000000	0.108469	1.03E+06	1.03E+06	∞	∞
perold	0.000000	0.000943	703,824	2.64E+06	∞	∞
pilot	0.000000	0.000290	26,633	30,427	∞	∞
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	∞	∞
pilot87	0.000000	0.000000	111,163	138,736	∞	∞
pilotnov	0.000000	0.000750	2.36E+07	1.35E+08	∞	∞
qap8	0.000000	4.000000	17,248	17,248	∞	∞
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	∞	∞

TABLE 4.1 (*cont.*)

Problem	$\rho_P(d)$	$\rho_D(d)$	$\|d\|$ Lower bound	$\|d\|$ Upper bound	$\log C(d)$ Lower bound	$\log C(d)$ Upper bound
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
truss	0.518928	10.000000	154,676	154,676	5.5	5.5
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 4.2

Summary statistics of distances to ill-posedness for the NETLIB suite (prior to preprocessing by CPLEX 7.1*).*

		$\rho_D(d)$ 0	$\rho_D(d)$ Finite	$\rho_D(d)$ ∞	Totals
$\rho_P(d)$	0	19	41	1	61
	Finite	2	24	2	28
	∞	0	0	0	0
	Totals	21	65	3	89

TABLE 4.3

Evaluation of ill-posedness of the 61 *primal ill-posed instances in the NETLIB suite (prior to preprocessing by CPLEX* 7.1*).*

Indication	Number of instances
Dependent equations	21
Implied reverse inequalities	40
Total	61

TABLE 4.4
Condition measures for the NETLIB suite after preprocessing 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
fit2d	317.000000	∞	1.90E+06	2.24E+06	3.8	3.8
fit2p	1.057333	1.000000	621,470	658,700	5.8	5.8
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	∞	∞
maros-r7	1.000131	0.846743	9.39E+06	9.39E+06	7.0	7.0
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	0.000002	0.000290	35,379	35,379	10.2	10.2
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
pilot87	0.000000	0.000199	100,187	125,426	∞	∞
pilotnov	0.000000	0.001146	2.36E+07	1.32E+08	∞	∞
qap8	0.022222	2.000000	17,248	17,248	5.9	5.9
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

TABLE 4.4 (*cont.*)

Problem	$\rho_P(d)$	$\rho_D(d)$	$\|d\|$ Lower bound	$\|d\|$ Upper bound	$\log C(d)$ Lower bound	$\log C(d)$ Upper bound
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
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
truss	0.518928	10.000000	154,676	154,676	5.5	5.5
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 4.5

Summary statistics of distances to ill-posedness for the NETLIB suite after preprocessing by CPLEX 7.1.

		$\rho_D(d)$ 0	$\rho_D(d)$ Finite	$\rho_D(d)$ ∞	Totals
$\rho_P(d)$	0	1	15	0	16
	Finite	1	70	2	73
	∞	0	0	0	0
	Totals	2	85	2	89

from Table 4.5 that 19% (17/89) of the postprocessed problems in the NETLIB suite are ill-posed. In contrast to the original problems, the vast majority of postprocessed problems have finite condition measures, as the preprocessing heuristics are very effective at identifying and correcting many instances of implied reverse inequalities in addition to finding and eliminating linearly dependent equations. We also examined the optimal solutions of (3.3) for the 16 primal ill-posed postprocessed problems in the NETLIB suite in order to evaluate the causes of the ill-posedness among these

TABLE 4.6
Evaluation of ill-posedness of the 16 *primal ill-posed instances in the NETLIB suite after preprocessing by CPLEX* 7.1.

Indication	Number of instances
Dependent equations	0
Implied reverse inequalities	16
Total	16

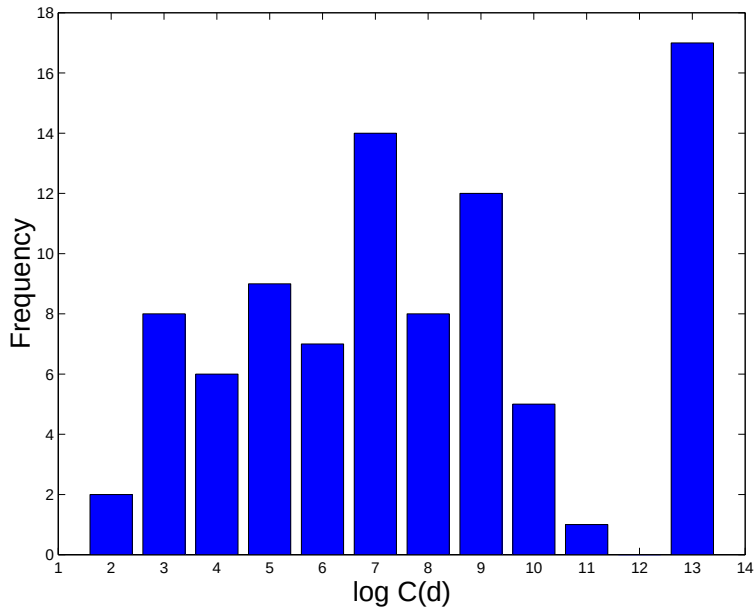

FIG. 4.1. *Histogram of condition measures for the NETLIB suite after preprocessing by CPLEX* 7.1 *(using the geometric mean of the lower and upper bound estimates of* $C(d)$*).*

postprocessed problem instances. Table 4.6 summarizes our findings, which show that all of the ill-posed postprocessed LP instances have implied reverse inequalities among the inequalities and/or lower/upper bounds.

Figure 4.1 presents a histogram of the condition measures of the postprocessed problems taken from Table 4.4. The condition measure of each problem is represented by the geometric mean of the upper and lower bound estimates in this histogram. The right-most 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 measure, $\log C(d)$ 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 nonfinite condition measure.

4.3. Condition measures and the observed performance of interior-point methods on the NETLIB suite. 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, [12].

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,

$$\vartheta := |L| + |G| + |L_B| + |U_B| - |L_B \cap U_B|, \tag{4.1}$$

and $\tilde{L}$ is the bit-size of a binary encoding of the problem instance data; see [23]. (Subtraction of the final term of (4.1) is shown in [7].) 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-measure 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 [25] for details. Of course, even here one must bear in mind that 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 (already small) can be accounted for by the condition measures of the problem instances that are the input to the IPM algorithm. The finite condition measures of the 72 postprocessed problems from the NETLIB suite shown in Table 4.4 provide a rich set of data that can be used to address this question. Here the goal is to assess whether or not condition measures are relevant for understanding the practical performance of IPM algorithms (we do *not* aim at validating the complexity theory).

In order to assess any relationship between condition measures and IPM iterations for the NETLIB suite, we first solved and recorded the IPM iterations for the 89 problems from the NETLIB suite. The problems were preprocessed 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 4.7. Notice that these iteration counts vary between 8 and 48.

Figure 4.2 shows a scatter plot of the number of IPM iterations taken by CPLEX 7.1 to solve the 89 problems in the NETLIB suite after preprocessing (from Table 4.7) and $\log C(d)$ of the postprocessed problems (using the $\log C(d)$ estimates from columns 6 and 7 of Table 4.4). In the figure, the horizontal lines represent the range for $\log C(d)$ due to the lower and upper estimates of $C(d)$ from the last two columns of Table 4.4. Also, similarly to Figure 4.1, problems with infinite condition measure are shown in the figure on the far right as a visual aid.

Figure 4.2 shows that as $\log C(d)$ 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 $\log C(d)$. The converse of this statement is not supported by the scatter plot: if a problem has a large value of $\log C(d)$, one cannot state in general that the problem will take a large number of IPM iterations to solve.

TABLE 4.7
IPM iterations for the NETLIB suite using CPLEX 7.1 *function* baropt.

Problem	IPM iterations	Problem	IPM iterations	Problem	IPM iterations
25fv47	22	gfrd-pnc	18	scorpion	13
80bau3b	30	greenbea	38	scrs8	20
adlittle	12	greenbeb	33	scsd1	10
afiro	9	grow15	12	scsd6	11
agg	22	grow22	12	scsd8	9
agg2	18	grow7	10	sctap1	13
agg3	21	israel	23	sctap2	15
bandm	16	kb2	17	sctap3	15
beaconfd	8	lotfi	14	share1b	22
blend	11	maros	27	share2b	14
bnl1	25	maros-r7	9	shell	16
bnl2	28	modszk1	23	ship04l	13
bore3d	16	perold	42	ship04s	17
brandy	19	pilot	22	ship08l	14
capri	19	pilot.ja	46	ship08s	14
cycle	25	pilot.we	48	ship12l	19
czprob	32	pilot4	35	ship12s	17
d2q06c	28	pilot87	26	sierra	16
d6cube	22	pilotnov	19	stair	16
degen2	13	qap8	9	standata	9
degen3	19	recipe	9	standgub	9
e226	18	sc105	10	standmps	13
etamacro	24	sc205	11	stocfor1	10
fffff800	30	sc50a	10	stocfor2	16
finnis	19	sc50b	9	truss	17
fit1d	14	scagr25	14	tuff	21
fit1p	13	scagr7	13	vtp.base	10
fit2d	18	scfxm1	18	wood1p	13
fit2p	18	scfxm2	20	woodw	21
ganges	13	scfxm3	20		

In order to be a bit more definitive, we ran a simple linear regression with the IPM iterations of the postprocessed problem as the dependent variable and $\log C(d)$ as the independent variable, for the 72 NETLIB problems which have a finite condition measure after preprocessing. For the purposes of the regression computation we used the geometric mean of the lower and upper estimates of the condition measure from the last two columns of Table 4.4. The resulting linear regression equation is

$$\text{IPM Iterations} = 4.1223 + 1.7490 \log C(d),$$

with $R^2 = 0.4160$. This indicates that in the sample of 72 NETLIB suite problem instances whose postprocessed condition measure is finite, about 42% of the variation in IPM iterations among these problems is accounted for by $\log C(d)$ of the postprocessed problem instance. A plot of this regression line is shown in Figure 4.3, where once again the 17 problems that are ill-posed 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 4.8.

The above regression analysis indicates that $\log C(d)$ accounts for 42% of the variation in IPM iteration counts among those NETLIB suite problem instances with finite postprocessed condition measure. However, recall that the complexity theory of interior-point methods bounds the number of IPM iterations by $O(\sqrt{\vartheta} \log(C(d)+\cdots))$. The factor $\sqrt{\vartheta}$ in the complexity bound seems to be a fixture of the theory of self-concordant barrier functions (see [14]), despite the belief that such dependence is not

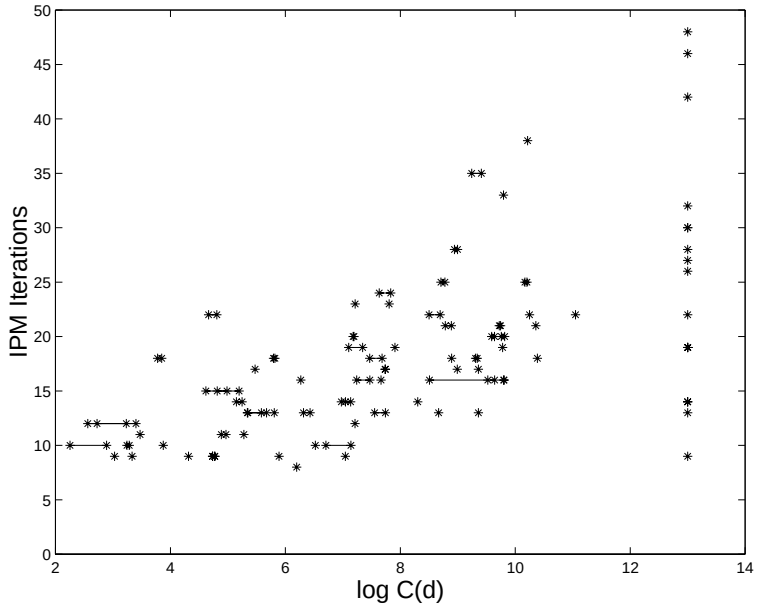

FIG. 4.2. *Scatter plot of IPM iterations and* $\log C(d)$ *for* 89 *NETLIB problems after preprocessing, using CPLEX* 7.1.

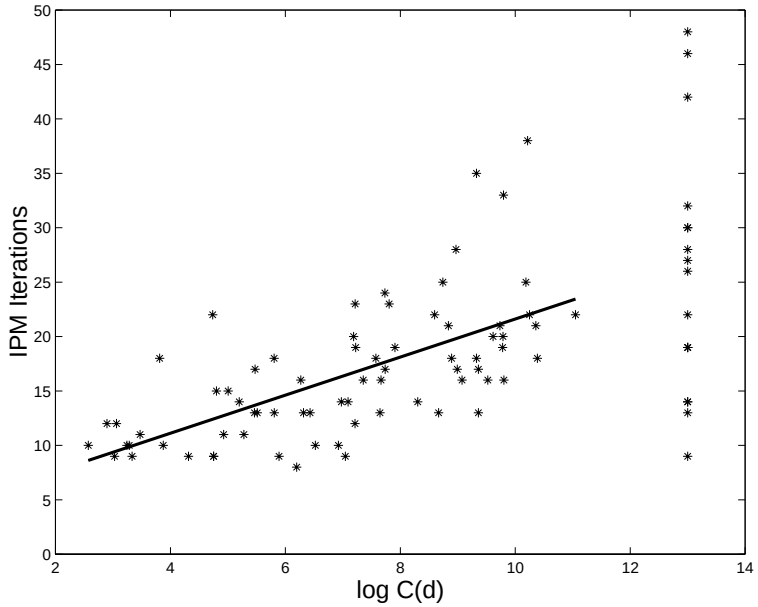

FIG. 4.3. *Linear regression of IPM iterations and* $\log C(d)$ *for* 72 *NETLIB problems with finite condition measure after preprocessing, using CPLEX* 7.1 *(using the geometric mean of the lower and upper bound estimates of* $C(d)$*).*

TABLE 4.8
Statistics for the linear regression of IPM iterations and $\log C(d)$.

Coefficient	Value	t-statistic	95% Confidence interval
β_0	4.1223	2.2480	[0.4650 , 7.7796]
β_1	1.7490	7.0620	[1.2551 , 2.2430]

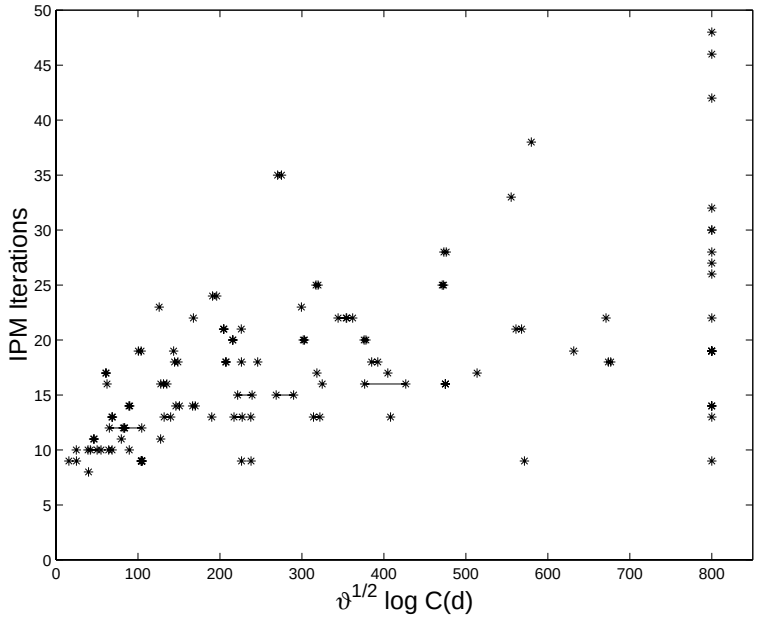

FIG. 4.4. *Scatter plot of IPM iterations and* $\sqrt{\vartheta}\log C(d)$ *for* 89 *NETLIB problems after preprocessing, using CPLEX* 7.1.

borne out in practice. Nevertheless, one can also ask whether $\sqrt{\vartheta}\log C(d)$ as opposed to $\log C(d)$ might better account for the variation in IPM iteration counts among the NETLIB suite problems. We now address this question. Figure 4.4 shows a scatter plot of the number of IPM iterations taken by CPLEX 7.1 to solve the 89 problems in the NETLIB suite after preprocessing and $\sqrt{\vartheta}\log C(d)$ of the postprocessed problems. (The horizontal lines refer to the range of the lower and upper estimates of $C(d)$ from the last two columns of Table 4.4; also, problems with infinite condition measure are shown in the figure on the far right as a visual aid.) We also ran a simple linear regression of IPM iterations of the postprocessed problem as the dependent variable and $\sqrt{\vartheta}\log C(d)$ as the independent variable, again for the 72 NETLIB problems which have a finite condition measure after preprocessing. The resulting linear regression equation is

$$\text{IPM Iterations} = 11.7903 + 0.0195\sqrt{\vartheta}\log C(d),$$

with $R^2 = 0.3021$. A plot of this regression is shown in Figure 4.5, and Table 4.9 shows the regression statistics. Notice that $R^2 = 0.3021$ for the $\sqrt{\vartheta}\log C(d)$ regression model, which is inferior to $R^2 = 0.4160$ for the $\log C(d)$ regression model. These results indicate that among the 72 NETLIB suite postprocessed problem instances with finite condition measure, $\log C(d)$ is better than $\sqrt{\vartheta}\log C(d)$ at accounting for the variation in IPM iterations for these NETLIB suite problems.

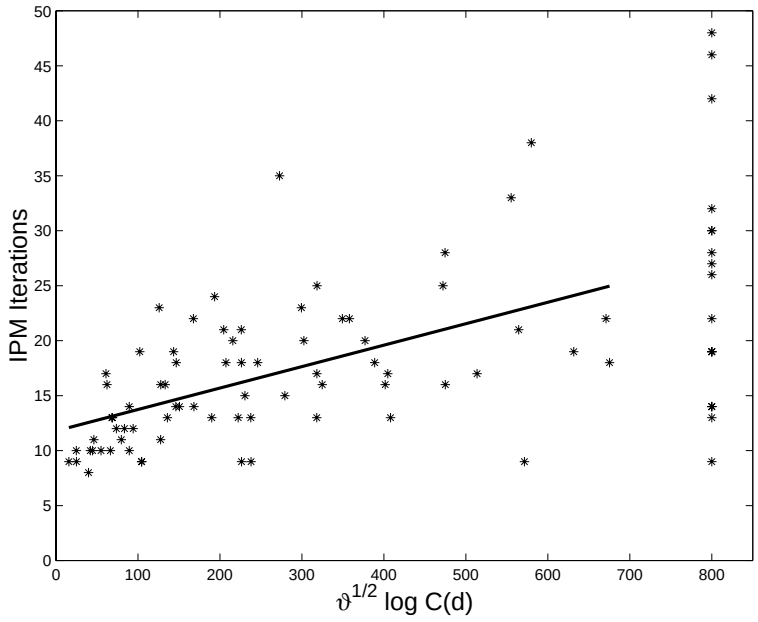

FIG. 4.5. *Linear regression of IPM iterations and* $\sqrt{\vartheta}\log C(d)$ *for* 72 *NETLIB problems with finite condition measure after preprocessing, using CPLEX* 7.1 *(using the geometric mean of the lower and upper bound estimates of* $C(d)$*).*

TABLE 4.9
Statistics for the linear regression of IPM iterations and $\sqrt{\vartheta}\log C(d)$*.*

Coefficient	Value	t-statistic	95% Confidence interval
β_0	11.7903	11.2667	[9.7031 , 13.8774]
β_1	0.0195	5.5046	[0.0124 , 0.0266]

TABLE 4.10
Sample correlations for 72 *NETLIB suite problems after preprocessing by CPLEX* 7.1 *(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					
$\log C(d)$	0.645	1.000				
$\log n$	0.383	0.217	1.000			
$\log m$	0.432	0.371	0.777	1.000		
$\log \vartheta$	0.398	0.224	0.991	0.808	1.000	
$\sqrt{\vartheta}$	0.311	0.093	0.909	0.669	0.918	1.000

We also computed the sample correlation coefficients of the IPM iterations from Table 4.7 with the following dimensional measures for the 72 problems in the NETLIB suite with finite condition measure of the postprocessed problem instance: $\log m$, $\log n$, $\log \vartheta$, and $\sqrt{\vartheta}$. The resulting sample correlations are shown in Table 4.10. Observe from Table 4.10 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 $R = 0.432$, and so a linear regression of IPM iterations as a function of $\log m$ would yield $R^2 = (0.432)^2 = 0.187$, which is decidedly less than $R^2 = 0.4160$ for $\log C(d)$. Also, note from Table 4.10 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 measures one necessarily fails to keep the problem size or structure invariant. Herein, we attempt to circumvent this 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 $(i^*, j^*, y^*, (s^+)^*, (s^-)^*, v^*)$ is an optimal solution of (3.14), then the rank-1 data perturbation

$$\Delta d = (\Delta A, \Delta b, \Delta c) := \big(-j^* e^{i^*} (A^t y^* + (s^+)^* - (s^-)^*)^t, -j^* e^{i^*} (b^t y^* - v^*), 0\big) \tag{4.2}$$

is a minimum-norm perturbation for which $\rho_P(d + \Delta d) = 0$ (where e^{i^*} denotes the (i^*)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 (4.2) 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 measure 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 preprocessing. According to Table 4.4, $\rho_D(\bar{d}) = 0.049075 \geq 0.021191 = \rho_P(\bar{d}) > 0$. Now let $\Delta\bar{d}$ be the perturbation of this data instance according to (4.2). 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 4.6. As the figure shows, the number of IPM iterations grows as the perturbed problem instance becomes more ill-posed.

The pattern of growth in IPM iterations as the perturbed problem becomes more ill-posed is not shared by all problem instances in the NETLIB suite. Figure 4.7 shows the plot of IPM iterations for problem `e226` as the perturbed problem instance becomes more ill-posed. For this problem instance the growth in IPM iterations is not monotone.

Of the 72 postprocessed problems in the NETLIB suite with finite condition measure, 59 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 postprocessed problem instance $\bar{d}_\alpha$. Then

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

is the difference between the IPM iterations needed to solve the unperturbed and fully perturbed problem instances. Table 4.11 shows some summary statistics of the

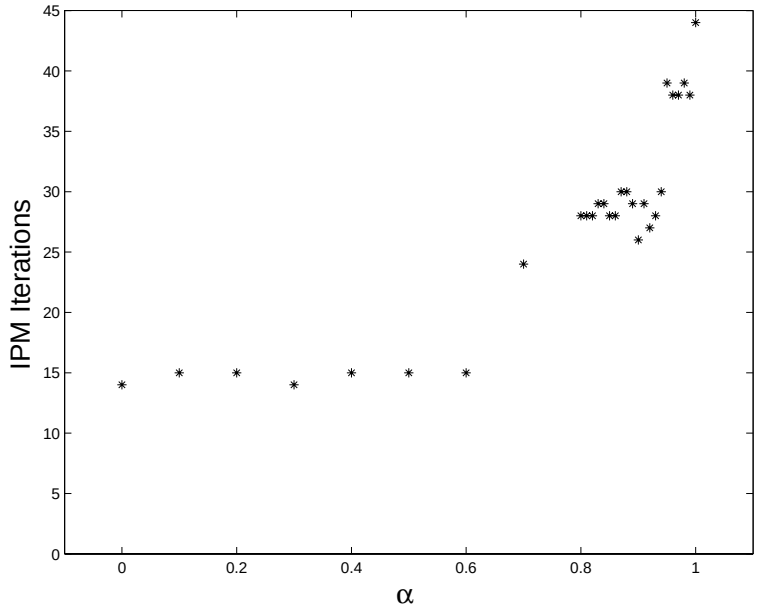

FIG. 4.6. *The number of IPM iterations needed to solve the perturbed postprocessed problem instance* `scagr25`*, as a function of the perturbation scalar* α*.*

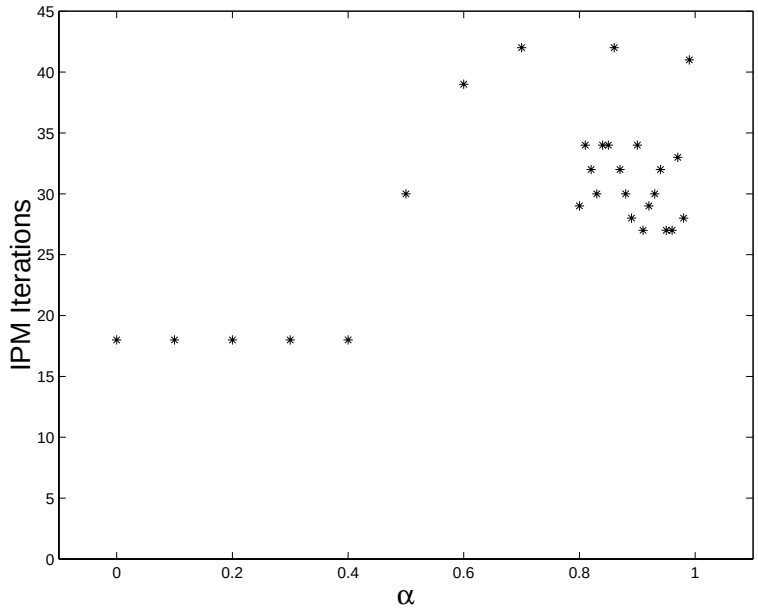

FIG. 4.7. *The number of IPM iterations needed to solve the perturbed postprocessed problem instance* `e226`*, as a function of the perturbation scalar* α*.*

distribution of Δk for the 59 problems in the NETLIB suite that are readily amenable to this analysis. As the table shows, the fully perturbed problem instance has a larger IPM iteration count in 68% (40 out of 59) of the problem instances. Curiously, the

TABLE 4.11
The distribution of the change in IPM iterations needed to solve the unperturbed problem instance and the fully perturbed problem instance for 59 postprocessed problems in the NETLIB suite.

Change in IPM iterations (Δk)	Number of problem instances
−1 or less	11
0	8
1 to 5	13
6 to 10	9
11 or more	18
Total	59

number of IPM iterations is actually less for the fully perturbed problem instance in 19% (11 out of 59) problem instances amenable to this analysis. A rough summary of the results in Table 4.11 is that the number of IPM iterations for the fully perturbed problem increases dramatically (more than 10 iterations) on 31% of the problem instances, increases modestly (1–10 iterations) on 37% of the problem instances, and remains the same or decreases slightly on 32% of problem instances.

5. Discussion and open questions. The purpose of this paper has been to study condition measures 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 measure computation and analysis, and we computed condition measures for 89 original NETLIB suite problem instances, as well as for the corresponding problem instances after preprocessing by CPLEX 7.1. We then investigated the extent to which the condition measure provides some explanatory value for the (already small) variance in the observed IPM iterations among problem instances in the NETLIB suite.

Except for certain classes of structured LP problems (see [12]), there is not yet a clear practical understanding (nor a theory) to explain the variation in the iteration counts of IPM algorithms (either theoretical or practical) on different LP instances. Herein we have explored the extent to which condition measures provide explanatory value for this variation. The scatter-plot in Figure 4.2 indicates that problem instances with large IPM iteration counts must have large condition measures. However, the converse of this assertion is not supported by the data; there are problem instances that have a high condition measure and low IPM iteration counts, for example, `agg2`, `recipe`, and some of the controlled-perturbation instances of section 4.4 with large condition measure.

It is easy to construct families of LP instances with ever-larger condition measures, whose IPM iteration counts do not grow excessively (see section 4.4). However, despite much effort, we have been unable to construct a family of problem instances with ever-larger practical IPM iteration counts but whose condition measures remains bounded. The existence of such a family is an open question.

The scatter-plot in Figure 4.2 indicates visually that there is a linear relationship between $\log C(d)$ and IPM iterations, and such a relationship is borne out by simple linear regression, with a resulting $R^2 = 0.4160$. However, in performing the regression analysis, there was no satisfactory way to include the 17 data instances with nonfinite (postprocessed) condition measures, and so these were removed, arguably biasing the results in favor of the explanatory value of the condition measure. A similar criticism can be made for the sample correlation coefficients computed in Table 4.10.

However, we feel that, at least on a relative basis, the results in Table 4.10 point to the conclusion that the condition measure does a better job of explaining the variation in IPM iteration counts than do any of the obvious reasonable alternative measures of problem size: $\log n$, $\log m$, $\log \vartheta$, or $\sqrt{\vartheta}$.

This work is a first attempt at explaining the observed performance of modern IPM solvers using condition measures that arise in the worst-case complexity analysis of interior-point methods. There are a variety of other instance-specific measures that have been used to bound the theoretical complexity of interior-point methods for linear optimization, including the bit-size L (see Karmarkar [11]), $\bar{\chi}_A$ (see Vavasis and Ye [26]), σ (see Ye [31]), and g and D_ϵ [6]. One natural question to ask is whether these or perhaps other measures might further explain the observed performance of IPM solvers.

Finally, the theory of condition measures referenced herein pertains to the very general class of conic convex optimization problems (and some formats for nonconic convex optimization problems as well), including semidefinite programming (SDP). Given the importance of SDP and the continuing development of IPM software for SDP, it is natural to ask to what extent condition measures (or other measures) might explain the observed performance of IPM solvers for SDP.

REFERENCES

[1] A. Ben-Tal and A. Nemirovski, *Robust solutions of linear programming problems contaminated with uncertain data*, Math. Program., 88 (2000), pp. 411–424.

[2] F. Cucker and J. Peña, *A primal-dual algorithm for solving polyhedral conic systems with a finite-precision machine*, SIAM J. Optim., 12 (2001), pp. 522–554.

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

[4] S. Filipowski, *On the complexity of solving sparse symmetric linear programs specified with approximate data*, Math. Oper. Res., 22 (1997), pp. 769–792.

[5] S. Filipowski, *On the complexity of solving feasible linear programs specified with approximate data*, SIAM J. Optim., 9 (1999), pp. 1010–1040.

[6] R. M. Freund, *Complexity of convex optimization using geometry-based measures and a reference point*, Math. Program., to appear.

[7] R. M. Freund and M. J. Todd, *Barrier functions and interior-point algorithms for linear programming with zero-, one-, or two-sided bounds on the variables*, Math. Oper. Res., 20 (1995), pp. 415–440.

[8] R. M. Freund and J. R. Vera, *Condition-based complexity of convex optimization in conic linear form via the ellipsoid algorithm*, SIAM J. Optim., 10 (1999), pp. 155–176.

[9] R. M. Freund and J. R. Vera, *On the complexity of computing estimates of condition measures of a conic linear system*, Math. Oper. Res., to appear.

[10] 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*, Math. Program., 86 (1999), pp. 225–260.

[11] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395.

[12] I. J. Lustig, R. E. Marsten, and D. F. Shanno, *The primal-dual interior point method on the Cray supercomputer*, in Large-Scale Numerical Optimization (Workshop held at Cornell University, Ithaca, NY, 1989), SIAM Proc. Appl. Math. 46, T. F. Coleman and Y. Li, eds., SIAM, Philadelphia, 1990, pp. 70–80.

[13] I. Lustig, R. Marsten, and D. Shanno, *Interior point methods: Computational state of the art*, ORSA J. Comput., 6 (1994), pp. 1–14.

[14] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM Stud. Appl. Math. 13, SIAM, Philadelphia, 1994.

[15] *NETLIB linear programming library*, online at http://www.netlib.org/lp/.

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

[17] M. A. Nunez and R. M. Freund, *Condition-measure bounds on the behavior of the central trajectory of a semidefinite program*, SIAM J. Optim., 11 (2001), pp. 818–836.
[18] F. Ordóñez, *On the Explanatory Value of Condition Numbers for Convex Optimization: Theoretical Issues and Computational Experience*, Ph.D. thesis, MIT, Cambridge, MA, 2002.
[19] J. Peña, *A characterization of the distance to infeasibility under structured perturbations*, Linear Algebra Appl., to appear.
[20] J. Peña, *Computing the Distance to Infeasibility: Theoretical and Practical Issues*, Technical report, Center for Applied Mathematics, Cornell University, Ithaca, NY, 1998.
[21] J. Peña, *Understanding the geometry of infeasible perturbations of a conic linear system*, SIAM J. Optim., 10 (2000), pp. 534–550.
[22] J. Peña and J. Renegar, *Computing approximate solutions for convex conic systems of constraints*, Math. Program., 87 (2000), pp. 351–383.
[23] J. Renegar, *A polynomial-time algorithm, based on Newton's method, for linear programming*, Math. Programming, 40 (1988), pp. 59–93.
[24] J. Renegar, *Some perturbation theory for linear programming*, Math. Programming, 65 (1994), pp. 73–91.
[25] J. Renegar, *Linear programming, complexity theory, and elementary functional analysis*, Math. Programming, 70 (1995), pp. 279–351.
[26] S. A. Vavasis and Y. Ye, *A primal-dual interior point method whose running time depends only on the constraint matrix*, Math. Programming, 74 (1996), pp. 79–120.
[27] J. R. Vera, *Ill-Posedness and the Computation of Solutions to Linear Programs with Approximate Data*, Technical report, Cornell University, Ithaca, NY, 1992.
[28] J. R. Vera, *Ill-Posedness in Mathematical Programming and Problem Solving with Approximate Data*, Ph.D. thesis, Cornell University, Ithaca, NY, 1992.
[29] J. R. Vera, *Ill-posedness and the complexity of deciding existence of solutions to linear programs*, SIAM J. Optim., 6 (1996), pp. 549–569.
[30] J. R. Vera, *On the complexity of linear programming under finite precision arithmetic*, Math. Programming, 80 (1998), pp. 91–123.
[31] Y. Ye, *Toward probabilistic analysis of interior-point algorithms for linear programming*, Math. Oper. Res., 19 (1994), pp. 38–52.