Summary Conclusions on Computational Experience and the Explanatory Value of Condition Measures for Linear Optimization*

Fernando Ordóñez† and Robert M. Freund‡
†*USC, Industrial and Systems Engineering Department, Los Angeles, CA*
‡*MIT Sloan School of Management, Cambridge, MA*

ABSTRACT

The modern theory of condition measures for convex optimization problems was initially developed for convex problems in the following conic format $(CP_d) : z_* := \min_x\{c^t x \mid Ax - b \in C_Y, x \in C_X\}$*, and several aspects of the theory have now been extended to handle non-conic 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 IPM solver, we also computed condition measures for the NETLIB suite problems after pre-preprocessing by CPLEX 7.1. Here we found that* 19% *of the post-processed problem instances in the NETLIB suite have infinite condition measure, and that* $\log C(d)$ *of the post-processed problems is fairly nicely distributed. Furthermore, there is a positive linear relationship between IPM iterations and* $\log C(d)$ *of the post-processed problem instances (significant at the* 95% *confidence level), and* 42% *of the variation in IPM iterations among the NETLIB suite problem instances is accounted for by* $\log C(d)$ *of the post-processed problem instances.*

I. INTRODUCTION

The modern theory of condition measures for convex optimization problems was initially developed in [22] for problems in the following conic format:

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

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 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 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):

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

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

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

and is defined similarly to $\rho_P(d)$ but using the dual problem instead. The quantity $C(d)$ is called the "condition 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:

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

and 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

*This paper contains substantial material from the full-length paper [16] "Computational Experience and the Explanatory Value of Condition Measures for Linear Optimization" by the same authors, which is under review for possible publication in *SIAM Journal on Optimization*. This research has been partially supported through the Singapore-MIT Alliance.

on changes in optimal objective values under perturbation, and numerical bounds related to the linear algebra computations of certain algorithms, see [22], [4], [3], [6], [7], [8], [26], [24], [27], [25], [18], [20]. 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 [13] and [14]. The connection of these condition measures to the complexity of algorithms has been shown in [6], [7], [23], [1], and [2], and some of the references contained therein. While this literature has focused almost exclusively on the conic format of (1), there have been some attempts to extend the theory to convex problems in structured non-conic formats, see Filipowski [3], Peña [19] and [17], and [15].

Given the theoretical importance of these many results, it is natural to ask what are typical values of these condition measures that arise in practice? Are such problems typically well-posed or ill-posed?, and how are the condition measures of such problems distributed? We begin to answer these question in this paper, where we compute and analyze these three condition measures for the NETLIB suite of industrial and academic 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 III-A.

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 [10], 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 [23] 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 measures of the LP problems that are solved? In order to answer this question, first note that typical IPM solvers perform routine pre-processing 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 pre-preprocessing by CPLEX 7.1. We found that 19% of the post-processed problem instances in the NETLIB suite have infinite condition measure, and that $\log C(d)$ of the post-processed problems is fairly nicely distributed, see Section III-B. In Section III-C, 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 instances, significant at the 95% confidence level, with $R^2 = 0.4160$. Therefore, about 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.

II. Linear Programming, Ground-Set Format, and Computation of $\rho_P(d)$, $\rho_D(d)$, and $C(d)$

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 [12]. Linear optimization problem instances arising in practice are typically conveyed in the following format:

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

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 "$b_{i,l} \le A_i x \le b_{i,u}$" 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 $\{0\}$ and by defining

$$P := \{x \mid x_j \ge l_j \text{ for } j \in L_B, \ x_j \le u_j \text{ for } j \in U_B\} \ , \tag{5}$$

we can naturally consider (4) to be an instance of the following "ground-set model" format:

$$(GP_d) \qquad \begin{array}{rll} z_*(d) = & \min & c^t x \\ & \text{s.t.} & Ax - b \in C_Y \\ & & x \in P, \end{array} \tag{6}$$

where P is called the ground-set. 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 (6), see [15]. We will use this ground-set format in our computation and evaluation of condition measures for LPs that arise in practice. The general set-up for the development of condition-measure theory for the ground-set model format is quite similar in thrust to development for problems in conic form reviewed in Section I. For details of the extension of condition-measure theory from conic form to the ground-set model format, see [15].

In [16] a methodology is described for computing $\rho_P(d)$ and $\rho_D(d)$ and for accurately estimating $C(d)$. The computational methodology is premised on the assumption that the norm on the space of the x variables in $I\!R^n$ 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 show in [16] how to compute $\rho_P(d)$ and $\rho_D(d)$ 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 $\rho_P(d)$ and $\rho_D(d)$ very much depends on the chosen norms, with the norms given as above being particularly appropriate for efficient computation of $\rho_P(d)$ and $\rho_D(d)$.

III. Computational Results on the NETLIB Suite of Linear Optimization Problems prior to Pre-Processing

A. Condition Measures for the NETLIB Suite prior to Pre-Processing

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. 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, and the sizes and diversity of the problems contained therein seem to be representative of 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 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 (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 $2n+2m$ 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 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 I shows summary statistics of 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 $\|d\|$ developed in [16]. All linear programming computation was performed using CPLEX 7.1 (function *primopt*). 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, that almost all of these (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.

TABLE I
Summary Statistics of Distances to Ill-Posedness for NETLIB Suite (prior to Pre-Processing by CPLEX 7.1).

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

The computational results in Table I have shown that 61 of the 89 LPs 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 LPs, $\rho_P(d) = 0$ can happen only if (i) there are linear dependencies among the equations of the problem instance (4), or (ii) if there is an implied reverse inequality among the inequalities and lower and upper bounds of the problem instance. A careful review of these problems shows that for at least 34% of the primal ill-posed problem instances (21 of the 61 problems), there are linear dependencies among the equations of (4).

B. Condition Measures 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 checks for eliminating linearly dependent equations, heuristics for identifying and eliminating redundant variable lower and upper bounds, and rules for row and/or column re-scaling, etc. The purposes of the pre-processing 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 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 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 pre-preprocessing by CPLEX 7.1. The processing used was the default CPLEX pre-processing with the linear dependency check option activated. Table II presents some summary statistics of these condition measures. Notice from Table II that 19% $(17/89)$ of the post-processed problems in the NETLIB suite are ill-posed. In contrast to the original problems, the vast majority of post-processed problems have finite condition measures, as the pre-processing 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 causes of ill-posedness for the 16 primal ill-posed post-processed problems in the NETLIB suite; we found that all of the ill-posed post-processed LP instances have implied reverse inequalities among the inequalities and/or lower/upper bounds.

TABLE II
SUMMARY STATISTICS OF DISTANCES TO ILL-POSEDNESS FOR THE NETLIB SUITE AFTER PRE-PROCESSING BY CPLEX 7.1.

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

Figure 1 presents a histogram of the condition measures of the post-processed problems. 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 non-finite condition measure.

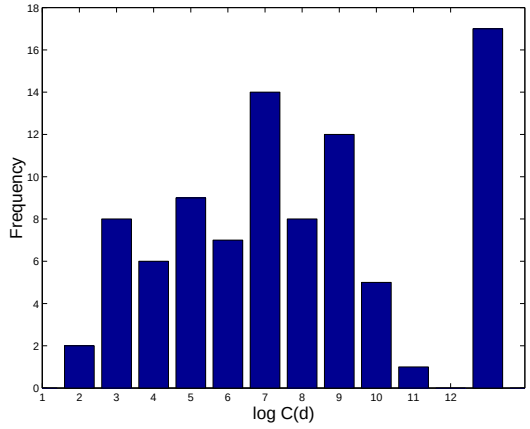

Fig. 1. Histogram of Condition Measures for the NETLIB Suite After Pre-Processing by CPLEX 7.1 (using the geometric mean of the lower and upper bound estimates of $C(d)$)

C. 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 [9].

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:

$$\vartheta := |L| + |G| + |L_B| + |U_B| - |L_B \cap U_B| \ , \qquad (7)$$

and $\tilde{L}$ is the bit-size of a binary encoding of the problem instance data, see [21] (subtraction of the final term of (7) 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-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 [23] 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 (albeit 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 post-processed problems from the NETLIB suite computed in this study 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 (and is *not* aimed 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 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}.

Figure 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 pre-processing and $\log C(d)$ of the post-processed problems. In the figure, we used the geometric mean of the lower and upper estimates of the condition measure $\log C(d)$. Also, similar to Figure 1, problems with infinite condition measure are shown in the figure on the far right as a visual aid. Figure 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.

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 $\log C(d)$ as the independent variable, for the 72 NETLIB problems which have a finite condition measure after pre-processing. For the purposes of the regression computation we used the geometric mean of the lower and upper estimates of the condition measure. The resulting linear regression equation is:

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

with $R^2 = 0.4160$. This indicates that 42% of the variation in IPM iteration counts among the NETLIB suite problems is accounted for by $\log C(d)$. A plot of this regression line is also shown in Figure 2, where once again the 17 problems that were 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 [16].

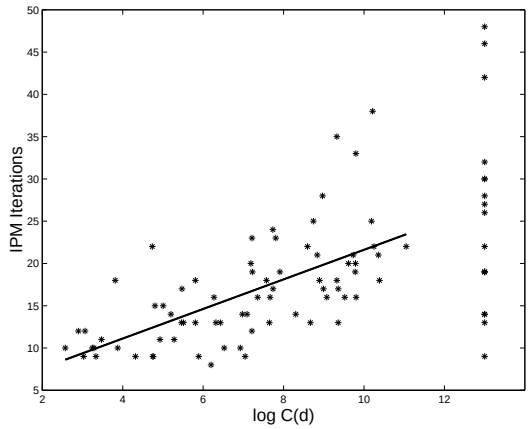

Fig. 2. Linear regression of IPM iterations and $\log C(d)$ for 72 NETLIB problems with finite condition measure after pre-processing, using CPLEX 7.1 (using the geometric mean of the lower and upper bound estimates of $C(d)$)

The above regression analysis indicates that $\log C(d)$ accounts for 42% of the variation in IPM iteration counts among the NETLIB suite of linear optimization problems.

We also ran a simple linear regression model of IPM iterations as the dependent variable and $\sqrt{\vartheta} \log C(d)$ of the post-processed problem instance as the independent variable. This yielded an inferior regression model, with $R^2 = 0.3021$. These results indicate that $\log C(d)$ explains better than $\sqrt{\vartheta} \log C(d)$ the variation in IPM iteration counts among the NETLIB suite of linear optimization instances.

We also computed the sample correlation coefficients of the IPM iterations with the following dimensional measures for the 72 problems in the NETLIB suite with finite condition measure of the post-processed problem instance:

TABLE III

SAMPLE CORRELATIONS FOR 72 NETLIB SUITE PROBLEMS AFTER PRE-PROCESSING BY CPLEX 7.1 (USING THE GEOMETRIC MEAN OF THE LOWER AND UPPER BOUND ESTIMATES OF $C(d)$).

	IPM iterations	$\log C(d)$	$\log n$	$\log m$	$\log \vartheta$
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

$\log m$, $\log n$, $\log \vartheta$, and $\sqrt{\vartheta}$. The resulting sample correlations are shown in Table III. Observe from Table III 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 III that both $\log \vartheta$ and $\sqrt{\vartheta}$ by themselves are significantly less correlated with the IPM iterations than $\log C(d)$.

IV. SUMMARY CONCLUSIONS

The purpose of this paper has been to gain some computational experience and to test the practical relevance of 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 pre-processing by CPLEX 7.1. This computation was done using the ground-set model format of convex optimization, where the ground-set was defined by the lower and upper bound constraints on the variables.

A summary of our computational findings is as follows:

1) 71% of the original problem instances in the NETLIB suite are ill-posed.
2) 69% 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. Among these primal ill-posed instances:
   - At least 34% of these instances exhibit linearly dependent equations.
   - At least 66% of these instances exhibit implied reverse inequalities.
3) After routine pre-processing by CPLEX 7.1, only 19% $(17/89)$ of problem instances are ill-posed. All of the 17 ill-posed instances have implied reverse inequalities among the inequalities and/or lower/upper bounds.
4) $\log C(d)$ of the 72 post-processed problems with finite condition measure 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 are related to the condition measures 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 measure. However, the converse of this statement is not supported by computational experience: if the post-processed problem instance has a large condition measure, 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 $\log C(d)$ of the post-processed problem instance as the independent variable yields a positive linear relationship between IPM iterations and $\log C(d)$, significant at the 95% confidence level, with $R^2 = 0.4160$. This means that 42% of the variation in IPM iterations among the NETLIB suite problems is accounted for by $\log C(d)$.
7) A simple linear regression model of IPM iterations as the dependent variable and $\sqrt{\vartheta}\log C(d)$ of the post-processed problem instance as the independent variable yields an inferior regression model, with $R^2 = 0.3021$. These results indicate that $\log C(d)$ explains better than $\sqrt{\vartheta}\log C(d)$ 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) In [16], we show that in controlled perturbations of problem instances to ill-posed 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] 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.

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] F. Ordóñez and R. M. Freund. Computational Experience and the Explanatory Value of of Condition Measures for Linear Optimization. Technical Report, Operations Research Center, MIT, August 2002.

[17] J. Peña. A characterization of the distance to infeasibility under structured perturbations. Working paper, GSIA, Carnegie Mellon University, 2002.

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

[19] J. Peña. Understanding the geometry of infeasible perturbations of a conic linear system. *SIAM Journal on Optimization*, 10(2):534–550, 2000.

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

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

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

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

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

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

[26] 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.

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