NOTE TO USERS

This reproduction is the best copy available.

UMI®
RICE UNIVERSITY

On Improving the Accuracy of Primal-Dual Interior Point Methods for Linear Programming

by

Shana Wang

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Master of Arts

APPROVED, THESIS COMMITTEE:

Yin Zhang, Chairman
Professor of Computational and Applied Mathematics

Richard A. Tapia
Noah G. Harding Professor of
Computational and Applied Mathematics

Mark Embree
Assistant Professor of Computational and Applied Mathematics

Houston, Texas

September, 2004
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI

UMI Microform 1425886
Copyright 2005 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346
Abstract

On Improving the Accuracy of Primal-Dual Interior Point Methods for Linear Programming

by

Shana Wang

Implementations of the primal-dual approach in solving linear programming problems still face issues in maintaining numerical stability and in attaining high accuracy. The major source of numerical problems occurs during the solving of a highly ill-conditioned linear system within the algorithm. We perform a numerical investigation to better understand the numerical behavior related to the solution accuracy of an implementation of an infeasible primal-dual interior-point (IPDIP) algorithm in LIPSOL, a linear programming solver. From our study, we learned that most test problems can achieve higher than the standard $10^{-8}$ accuracy used in practice, and a high condition number of the ill-conditioned coefficient matrix does not solely determine the attainable solution accuracy. Furthermore, we learned that the convergence of the primal residual is usually most affected by numerical errors. Most importantly, early satisfaction of the primal equality constraints is often conducive to eventually achieving high solution accuracy.
Acknowledgements

I would like to first express my deepest gratitude and sincere appreciation to my advisor, Dr. Yin Zhang. Without him, this thesis would not be possible. Special thanks to Dr. Richard Tapia and Dr. Mark Embree for serving on my committee. Many thanks to Daria for keeping me on my toes. Finally, I would like to thank my husband, David Lee, for all of his support and encouragement.
Contents

Abstract ii

Acknowledgements iii

List of Figures vi

List of Tables vii

1 Introduction to Linear Programming 1
   1.1 Primal and Dual Programs .......................... 1
   1.2 Duality and Optimality ............................ 3
   1.3 Linear Programming Solution Approaches .......... 4

2 Infeasible Primal-Dual Interior Point Methods 6
   2.1 Primal-Dual Methods .................................. 6
   2.2 Newton’s Method ..................................... 7
   2.3 Path-Following on the Central Path ................. 8
   2.4 IPDIP Framework ..................................... 9
   2.5 Implementation Details ............................. 11
      2.5.1 Centering and Step-Length Parameters .......... 11
      2.5.2 Stopping Criterion ................................ 13

3 Numerical Stability of the IPDIP Algorithm 14
3.1 Solving the Linear System ........................................ 14
3.2 Ill-Conditioning ........................................ 16

4 Solution Accuracy of the IPDIP Algorithm .......................... 19
  4.1 Test Environment - LIPSOL .................................... 19
  4.2 Test Problems - Netlib ....................................... 20
  4.3 Solution Accuracy Under Different Tolerances .................. 21
  4.4 A Look at Condition Numbers ................................. 23
  4.5 Primal Residual ........................................ 24
  4.6 Data Truncation ........................................ 28

5 Conclusions and Future Work .................................. 36
  5.1 Conclusions ........................................ 36
  5.2 Some Experiments: Room for Future Work .................... 37
    5.2.1 Early Satisfaction of Primal Feasibility ................ 37
    5.2.2 Reduce the RHS Error .................................. 38
    5.2.3 Other Experiments ..................................... 38
  5.3 Final Remarks ........................................ 39

A LIPSOL Results from Solving Netlib Problems .................... 40

Bibliography ........................................ 45
List of Figures

4.1 Condition Numbers for "Easy" Problems 1 ........................................ 26
4.2 Condition Numbers for "Easy" Problems 2 ........................................ 27
4.3 Condition Numbers for "Easy" Problems 3 ........................................ 28
4.4 Condition Numbers for "Hard" Problems 1 ........................................ 29
4.5 Condition Numbers for "Hard" Problems 2 ........................................ 30
4.6 Residual Behavior for df1001 ............................................................. 31
4.7 Residual Behavior for seba ................................................................. 32
4.8 Residual Behavior for grow7 ............................................................... 33
4.9 Residual Behavior for fit2 Problems .................................................... 34
4.10 Biggest Order of Primal Residual Decrease for All Iterations .......... 35
List of Tables

4.1 Problems Meeting the Convergence Criterion for High Accuracy . . . 22
4.2 Condition Numbers for “Easy” Problems . . . . . . . . . . . . . . . . . 24
4.3 Condition Numbers for “Hard” Problems . . . . . . . . . . . . . . . . . 25

A.1 Test Problems: Control Group (I) . . . . . . . . . . . . . . . . . . . . . 41
A.2 Test Problems: Control Group (II) . . . . . . . . . . . . . . . . . . . 42
A.3 Test Problems: Control Group (III) . . . . . . . . . . . . . . . . . . . . 43
A.4 Test Problems: Control Group (IV) . . . . . . . . . . . . . . . . . . . . 44
Chapter 1

Introduction to Linear Programming

In this chapter, we introduce the basic concepts of linear programming and present its important optimality results. We conclude by reviewing the main approaches used in solving linear programs.

1.1 Primal and Dual Programs

A Linear Program (LP) is a mathematical problem in which a linear objective function is optimized subject to linear constraints. These constraints can be equalities or inequalities. We consider the following form of a primal LP\(^1\):

\[
\begin{align*}
\text{min} & \quad c^T x \\
\text{s.t.} & \quad Ax = b \\
& \quad 0 \leq x_i \leq u_i, \quad i \in \mathcal{I} \\
& \quad x_i \geq 0, \quad i \in \mathcal{J}
\end{align*}
\]

(1.1)

where \(c, x \in \mathbb{R}^n, b \in \mathbb{R}^m,\) and \(A \in \mathbb{R}^{m \times n}\). The index sets, \(\mathcal{I}\) and \(\mathcal{J}\), are such that \(\mathcal{I} \cup \mathcal{J} = \{1, 2, ..., n\}\) and \(\mathcal{I} \cap \mathcal{J} = \emptyset\). Without loss of generality, we let \(\mathcal{I} = \{1, 2, ..., n_u\}\)

\(^1\)We introduce the LP problem based on how it is defined and handled in LIPSOL [21].
and $J = \{n_{u+1}, n_{u+2}, \ldots, n\}$ for natural numbers $n_u \leq n$ such that $n_u$ is the cardinality of the set of $x_i$'s with upper bounds and $(n - n_u)$ is the cardinality of the set of $x_i$'s without upper bounds. We define $u_i \in \mathbb{R}^{n_u}$.

We convert Problem (1.1) into the LP standard form by adding a slack variable to the inequality constraint:

$$
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax = b \\
& \quad x_u + s = u \\
& \quad x, s \geq 0
\end{align*}
$$

(1.2)

where $x_u, s, u \in \mathbb{R}^{n_u}$. Moreover, $x_u$ is such that $[x_u]_i = \{x_i | i = 1, 2, \ldots, n_u\}$.

The dual of Problem (1.2) is

$$
\begin{align*}
\max & \quad b^T y - u^T w \\
\text{s.t.} & \quad A^T y + z - \text{app}(w) = c \\
& \quad w, z \geq 0
\end{align*}
$$

(1.3)

where $y \in \mathbb{R}^m$, $w \in \mathbb{R}^{n_u}$, and $z, \text{app}(w) \in \mathbb{R}^n$. The operator $\text{app}(\cdot)$ appends a vector of zeros in dimension $\mathbb{R}^{(n-n_u)}$ to vectors in $\mathbb{R}^{n_u}$ in order to expand the vector to a dimension of $\mathbb{R}^n$. Thus, for $w$,

$$
[\text{app}(w)]_i = \begin{cases} 
  w_i & \text{if } 1 \leq i \leq n_u \\
  0 & \text{if } n_u \leq i \leq n
\end{cases}
$$

The set of primal feasible solutions is:

$$
\mathcal{F}_p = \{ (x, s) \mid Ax = b, \ x_u + s = u, \ (x, s) \geq 0 \}
$$

(1.4)

If $(x, s) \in \mathcal{F}_p$, then the point is a primal feasible solution. If $(x, s) \notin \mathcal{F}_p$, then the point is not a primal feasible solution. Likewise, the set of dual feasible solutions is:

$$
\mathcal{F}_d = \{ (y, w, z) \mid A^T y + z - \text{app}(w) = c, \ (w, z) \geq 0 \}
$$

(1.5)

If $(y, w, z) \in \mathcal{F}_d$, then the point is a dual feasible solution. If $(y, w, z) \notin \mathcal{F}_d$, then the point is not a dual feasible solution. If $(x, s) \in \mathcal{F}_p$ and $(y, w, z) \in \mathcal{F}_d$, then $(x, s, y, w, z)$ is a primal-dual feasible solution.
1.2 Duality and Optimality

The primal and dual problems are more than just different representations of the same problem. They have a special relationship that can be explained by the following duality theorems.

**Theorem 1** The dual of the dual problem is the primal problem.

**Theorem 2 (Weak Duality)** If \((x, s) \in \mathcal{F}_p\) and \((y, w, z) \in \mathcal{F}_d\), then \((b^T y - u^T w) \leq c^T x\).

**Theorem 3 (Strong Duality)** A primal feasible point \((x^*, s^*)\) is optimal and a dual feasible point \((y^*, w^*, z^*)\) is optimal if and only if \((b^T y^* - u^T w^*) = c^T x^*\).

Theorem 1 shows that the primal and dual problems have a symmetric relationship. From Weak Duality, we see that the problems are bounded by each other when the solution is primal-dual feasible. Specifically, a feasible solution for the dual gives a lower bound on the primal while a feasible solution for the primal gives an upper bound on the dual. The difference between the feasible primal and dual solutions, \(c^T x - (b^T y - u^T w)\), is called the duality gap. By Weak Duality, we have \(c^T x - (b^T y - u^T w) \geq 0\). From Strong Duality, we see that the optimal objective values for the primal and dual problems are equal. Thus, the duality gap closes at optimality since \(c^T x - (b^T y - u^T w) = 0\).

When the \((x, s, y, w, z)\) is primal-dual feasible, the duality gap can be written as:

\[
x^T z + s^T w
\]

Thus, the weak duality can also be represented by \(x^T z + s^T w \geq 0\). Since \(x, z, s, \) and \(w\) have nonnegativity constraints, we can rewrite the Weak Duality as:

\[
x \circ z \geq 0 \quad (1.7)
\]

\[
s \circ w \geq 0 \quad (1.8)
\]
where $\circ$ represents component-wise multiplication.

As a consequence of Strong Duality, $x^T z + s^T w = 0 \Rightarrow x^T z = -(s^T w)$ at optimality. Since $x$, $z$, $s$, and $w$ are nonnegative, the following must occur at optimality:

\[
\begin{align*}
  x \circ z &= 0 \quad \text{(1.9)} \\
  s \circ w &= 0 \quad \text{(1.10)}
\end{align*}
\]

Equations (1.9) and (1.10) are known as the complementarity conditions for primal-dual optimality.

We can, thus, list the necessary and sufficient optimality conditions for the optimal solution to the primal and dual problems:

\[
\begin{align*}
  Ax &= b \quad \text{(1.11)} \\
  x_u + s &= u \quad \text{(1.12)} \\
  A^T y + z &= \text{app}(w) + c \quad \text{(1.13)} \\
  x \circ z &= 0 \quad \text{(1.14)} \\
  s \circ w &= 0 \quad \text{(1.15)} \\
  (x, z, s, w) &\geq 0 \quad \text{(1.16)}
\end{align*}
\]

These are also known as the First-Order Necessary Conditions or the Karush-Kuhn-Tucker (KKT) conditions.

1.3 Linear Programming Solution Approaches

There are two main classes of algorithmic approaches for solving Problem (1.2), the Simplex methods and the interior-point methods.

The first Simplex method was invented by George Dantzig [3] in 1947. Later, it developed into a class of methods that is still widely studied and used today [2]. Its solution path moves from vertex to adjacent vertex at each iteration, traveling along the edges of the feasible polyhedron until the optimal solution is either found
or determined not to exist. A primal Simplex method maintains primal feasibility and complementarity at all times, but the nonnegativity constraints for the dual problem are not satisfied until the optimal solution is found. From a practical standpoint, the Simplex Method is very successful in solving most problems due to its efficient and robust nature. In theory, there is no polynomial complexity bound available to this date for Simplex methods. Its performance has also been shown to be inferior to that of interior-point methods for some large-scale or highly degenerate problems [15]. Since Dantzig’s invention of the Simplex Method, improvements to the theory and practical implementation have continually been made, making it still one of the two major methods practically used in solving LP problems.

The interior-point methods, on the other hand, find an optimal solution by cutting through the inside of the feasible polyhedron without touching the boundaries. Its solution path is in the “respective” interior of the feasible set. In theory, many interior-point methods can find an approximate solution within a prescribed accuracy in polynomial time. In practice, the obtained solution will be close, but never exact, to the true solution. Unlike the Simplex Method, each iteration is expensive to compute but can make significant progress towards the solution.

It was not until Karmarkar’s 1984 paper discussing a polynomial-time algorithm [8] that interior-point methods received wide acclaim. His work sparked a research fervor that resulted in much theoretical and practical development in the past two decades. Several algorithmic variations of interior-point methods exist and have been successfully implemented, the most successful being the primal-dual approach. The primal-dual interior-point method solves the primal and dual LP problems simultaneously and will be the algorithm studied in this thesis.
Chapter 2

Infeasible Primal-Dual Interior Point Methods

In this chapter, we introduce an algorithmic framework for an infeasible primal-dual interior-point (IPDIP) method from a practical viewpoint. We review the basic concepts necessary to formulate the algorithm, present the IPDIP framework, and address some implementation details.

2.1 Primal-Dual Methods

The optimality conditions for Problem (1.2) can be expressed as a square system of linear-quadratic equations with non-negativity constraints [21]:

\[
F(x, s, w, z, y) = \begin{bmatrix}
Ax - b \\
x_u + s - u \\
A^Ty + z - \text{app}(w) - c \\
x \odot z \\
s \odot w
\end{bmatrix} = 0, \quad (x, s, w, z) \geq 0 \quad (2.1)
\]

Upon solving Equation (2.1), we simultaneously solve the primal and dual LP problems. This is known as the primal-dual method because it treats the primal
and dual LP problems equally while simultaneously solving them. The primal-dual feasible solution set for Equation (2.1) is:

$\mathcal{F} = \{(x, s, w, z, y) \mid Ax = b, x_u + s = u, ATy + z - \text{app}(w) = c, (x, s, w, z) \succeq 0\}

(2.2)

Most primal-dual interior-point methods require its nonnegatively-bounded iterates to be strictly feasible. In this case, the primal-dual strictly feasible set for Equation (2.1) is:

$\mathcal{F}^0 = \{(x, s, w, z, y) \mid Ax = b, x_u + s = u, ATy + z - \text{app}(w) = c, (x, s, w, z) > 0\}

(2.3)

2.2 Newton's Method

In the IPDIP algorithm, a modified Newton's Method is used to solve Equation (2.1). The solution to a square nonlinear system of equations, $F(q) = 0$, can be approximated by its first-order Taylor expansion:

$F(q) = 0 \approx F(q_0) + F'(q_0)(q - q_0)

(2.4)$

where the vector $q_0$ is a fixed point and $F'(\cdot)$ is the Jacobian of $F(\cdot)$. Recalling Equation (2.1), its Jacobian is:

$F'(x, s, w, z, y) = \begin{bmatrix}
A & 0 & 0 & 0 & 0 \\
\hat{I}^T & I_u & 0 & 0 & 0 \\
0 & 0 & -\hat{I} & I & AT \\
Z & 0 & 0 & X & 0 \\
0 & W & 0 & S & 0
\end{bmatrix},

(2.5)$

where $I_u \in \mathbb{R}^{(n_u \times n_u)}$ is an identity matrix and $\hat{I}^T = \begin{bmatrix} I_u & 0 \end{bmatrix} \in \mathbb{R}^{(n_u \times n)}$. Note that $X$, $S$, $W$, and $Z$ are, respectively, diag$(x)$, diag$(s)$, diag$(w)$, and diag$(z)$. 
Given an initial point, \( q_0 \), Newton’s Method computes an iteration sequence, \( \{q^k\} \), by the following recursive formula:

\[
q^{k+1} = q^k - [F'(q^k)]^{-1}F(q^k)
\]

(2.6)

There exist several improved modifications to the Newton’s Method, which enhances its convergence properties. It is well-known that the Newton’s method has good local convergence, but not global convergence, properties. The damped Newton’s method introduces a step-length parameter, \( \alpha^k > 0 \), at each iteration:

\[
q^{k+1} = q^k - \alpha^k[F'(q^k)]^{-1}F(q^k)
\]

(2.7)

When properly chosen, \( \alpha^k \) can be used to modify the step size at each iteration to enhance global convergence.

The composite Newton’s method improves the convergence rate of the Newton’s Method by using the same Jacobian matrix twice. Given the current iterate, \( q^k \), the following is computed at one iteration:

\[
q^{k+1} = q^k - F'(q^k)(F(q^k) - F(q_{tk}^k))^{-1}
\]

(2.8)

where \( q_{tk}^k \) is first computed to be:

\[
q_{tk}^k = q^k - F'(q^k)^{-1}F(q^k)
\]

(2.9)

In essence, the composite Newton’s method performs two, instead of one, back-solves per matrix factorization.

### 2.3 Path-Following on the Central Path

The central path is defined as:

\[
C = \{(x, s, y, w, z) \mid F(x, s, y, w, z) = \tau \varepsilon, \ (x, s, w, z) \geq 0\}
\]

(2.10)
where $\tau \geq 0$ and
\[
[e]_i = \begin{cases} 
0, & 1 \leq i \leq (m + n_u + n) \\
1, & (m + n_u + n) + 1 \leq i \leq (m + 2n_u + 2n)
\end{cases}
\]

Note that $[e]_i = 1$ for the complementarity components of $F_i(q)$, so the components of $x \circ z$ and $s \circ w$ are all required to have the same value. Also, note that as $\tau \to 0$, $\mathcal{C}$ approaches the solution. The central path plays an important role in guiding an LP algorithm to its solution. Instead of taking a pure Newton step, a primal-dual interior-point method can restrict the iterates to a neighborhood of the central path. In addition to following the central path to a solution, the method biases the step towards the interior of the nonnegative orthant, allowing for bigger steps to be taken without violating the nonnegativity constraints.

### 2.4 IPDIP Framework

In primal-dual interior-point methods, we attempt to solve a variation of Equation (2.6):
\[
F'(q^k)\Delta q^k = -F(q^k), \quad (x^k, s^k, w^k, z^k) \geq 0
\]
(2.11)

where $q^k = (x^k, s^k, w^k, z^k, y^k)$ and $\Delta q^k = (\Delta x^k, \Delta s^k, \Delta w^k, \Delta z^k, \Delta y^k)$.

We want to find the step, $\Delta q^k$, such that $(q^k + \Delta q^k)$ is a better approximation to the solution. This step is known as the pure Newton or “affine-scaling” [4] search direction.

A situation can occur whereby a nonnegatively-constrained variable is permanently set to zero when solving Equation (2.11). Strict feasibility must, therefore, be enforced, so $(x, s, w, z, y) \in \mathcal{F}^0$. Likewise, if a nonnegatively-constrained variable approaches zero much faster than others, it will not only have trouble recovering from such a small value, but will hinder a step’s potential length. Thus, we must additionally ensure that the complementarity pairs converge to zero at a similar rate. We
introduce the parameters $\alpha^k$ and $\mu^k$ to prevent the two aforementioned problematic occurrences:

$$q^{k+1} = q^k - \alpha^k F'(q^k)^{-1}(F(q^k) - \mu^k \hat{e})$$

(2.12)

where $(\alpha^k, \mu^k) > 0$.

Equation (2.12) can be described as the damped Newton’s method with a perturbation on the quadratic components of $F(q)$. The step-length parameter, $\alpha^k$, is chosen so that $[q^{k+1}]_i > 0$ for $1 \leq i \leq (2n_u + 2n)$, enforcing the nonnegative variable components to be strictly positive. The centering parameter, $\mu^k$, is controlled so that the complementarity pairs converge to zero at a similar pace.

In feasible interior-point methods, the initial point, $q_0$, must satisfy the primal, upper bound, and dual feasibilities, along with the positivity constraints. In infeasible interior-point methods, $q_0$ must only satisfy the positivity constraints [9].

We now introduce an algorithmic framework for an infeasible primal-dual interior-point (IPDIP) method:

<table>
<thead>
<tr>
<th>Algorithm 1 - IPDIP:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose $q^0$ such that $[q^0]_i \geq 0$ for $i \in [1, 2n_u + 2n]$</td>
</tr>
<tr>
<td>Set $k = 0$ and maxiter = 100</td>
</tr>
<tr>
<td><strong>WHILE</strong> (error &lt; tolerance) and ($k &lt; \text{maxiter}$)</td>
</tr>
<tr>
<td>1 Choose $\mu^k$</td>
</tr>
<tr>
<td>2 Compute $\Delta q^k = -F'(q^k)^{-1}[F(q^k) - \mu^k \hat{e}]$</td>
</tr>
<tr>
<td>3 Choose $\alpha^k$ such that $[q^{k+1}]_i \geq 0$ for $i \in [1, 2n_u + 2n]$</td>
</tr>
<tr>
<td>4 Compute $q^{k+1} = q^k + \alpha^k \Delta q^k$</td>
</tr>
<tr>
<td>5 $k = k + 1$</td>
</tr>
<tr>
<td><strong>END</strong></td>
</tr>
</tbody>
</table>

Essentially, using the idea of the composite Newton’s Method, Mehrotra [10] adds a corrector step to an initial “affine-scaling” search direction (predictor step) in order to obtain a second-order approximation to the solution. During the corrector step,
he also adaptively adjusts the centering parameter, $\mu^k$.

**Algorithm 2 - IPDIP with MPC:**

Choose $q^0$ such that $[q^0]_i \geq 0$ for $i \in [1, 2n_u + 2n]$

Set $k = 0$ and maxiter = 100

**WHILE** (error < tolerance) and ($k < \text{maxiter}$)

1. Compute $\Delta q^k_{\text{aff}} = -F'(q^k)^{-1}F(q^k)$

2. Choose $\mu^k$

3. Compute $\Delta q^k = -F'(q^k)^{-1}[F(q^k + \Delta q^k_{\text{aff}}) - \mu^k \hat{e}]$

4. Choose $\alpha^k$ such that $[q^{k+1}]_i \geq 0$ for $i \in [1, 2n_u + 2n]$

5. Compute $q^{k+1} = q^k + \alpha^k(\Delta q^k_{\text{aff}} + \Delta q^k)$

6. $k = k + 1$

**END**

Note that $\Delta q^k_{\text{aff}}$ is an "affine" step.

### 2.5 Implementation Details

In this section, we discuss implementation specifics for Algorithm 2, including the choice of the centering and step-length parameters, and establish an appropriate stopping criterion.

#### 2.5.1 Centering and Step-Length Parameters

The centering parameter, $\mu^k$, is composed of two components:

$$\mu^k = \sigma^k \theta^k$$  \hspace{1cm} (2.13)

where $\theta^k$, the duality measure, is defined as:

$$\theta^k(x^k, s^k, w^k, z^k) = \frac{[x^k]^T z^k + [s^k]^T w^k}{n + n_u}$$
and $\sigma^k \in (0, 1)$ is defined as:

$$\sigma^k = \left( \frac{x^T \hat{z} + s^T \hat{w}}{[x^k]^T z^k + [s^k]^T w^k} \right)^2$$

The parameter $\theta^k$ is simply the normalized duality gap, a reduction of which signifies progress towards the solution, and $\sigma^k$ determines a step's degree of centrality. When $\sigma^k = 0$, an “affine” step is computed. As $\sigma^k \to 1$, the search direction becomes more and more biased towards the central path.

In Step 2 of Algorithm 2, $\sigma^k$ is the component of $\mu^k$ that is adaptively modified after computing $\Delta q^k_{\text{aff}}$. Before computing $\sigma^k$, we must first find the step-length parameters, $(\hat{\alpha}_p, \hat{\alpha}_d)$, to determine the longest predictor-computed step length ($\Delta q^k_{\text{aff}}$) that can be taken in the primal and dual directions so the new iterate does not violate nonnegativity constraints. The intermediate iterates are:

$$\begin{aligned}
(\hat{x}, \hat{s}) &= (x^k, s^k) + \min(1, \hat{\alpha}_p)(\Delta x^k_{\text{aff}}, \Delta s^k_{\text{aff}}) \\
(\hat{w}, \hat{z}) &= (w^k, z^k) + \min(1, \hat{\alpha}_d)(\Delta w^k_{\text{aff}}, \Delta z^k_{\text{aff}})
\end{aligned}$$

which allows us to compute $\sigma^k$.

In Step 4 of Algorithm 2, the step-length parameter, $\alpha^k$, is also composed of two components:

$$\alpha^k = (\alpha_p, \alpha_d) = \tau_0(\hat{\alpha}_p, \hat{\alpha}_d) \quad (2.14)$$

where $(\hat{\alpha}_p, \hat{\alpha}_d)$ are similarly computed to determine the longest step lengths that can be taken in the primal and dual directions so $q^{k+1}$ does not violate the nonnegativity constraints. $\tau_0 \in [0.9, 1)$ is chosen to take a step back from the boundary to ensure that the new iterate falls within the neighborhood, $\mathcal{N}$, of the central path, $\mathcal{C}$. This is to ensure that the nonnegativity variables remain strictly positive. The neighborhood, $\mathcal{N}$, of $\mathcal{C}$ is defined as:

$$\mathcal{N} = \{q^k : (x^k, s^k, w^k, z^k) > 0, \min(x^k \circ z^k, s^k \circ w^k) \geq \phi \theta^k(x^k, s^k, w^k, z^k)\} \quad (2.15)$$

where $\phi \in [0, 1)$. Note that the set in (2.15) is called a one-sided $\infty$-norm neighborhood, which can encompass most of the feasible region, $\mathcal{F}$, if $\phi$ is close to zero [14].
After computing \((\alpha_p, \alpha_d)\), the new iterate will be:

\[
q^{k+1} = q^k + \begin{bmatrix}
\alpha_p \Delta x^k \\
\alpha_p \Delta s^k \\
\alpha_d \Delta w^k \\
\alpha_d \Delta z^k \\
\alpha_d \Delta y^k
\end{bmatrix}
\]  \hspace{1cm} (2.16)

### 2.5.2 Stopping Criterion

Since interior-point methods never reach the exact solution, we must establish a stopping criterion for judging when the iterates are sufficiently close to the solution.

The relative error\(^1\) of each component of our iterate is used to measure our proximity to the solution. We compare the total relative error (TRE) against the accuracy tolerance:

\[
\frac{\|r_p\|}{\max(1, \|b\|)} + \frac{\|r_d\|}{\max(1, \|c\|)} + \frac{\|r_u\|}{\max(1, \|u\|)} + \frac{|c^T x - b^T y + u^T w|}{\max(1, |c^T x|, |b^T y - u^T w|)} \leq \text{tolerance},
\]

where \(r_p^k, r_u^k,\) and \(r_d^k\) are, respectively, the primal, upper bound, and dual residuals. Note that the last term measures the relative residual of the duality gap. It is standard practice to set the stopping criterion tolerance to \(10^{-8}\) [19].

---

\(^1\)We use the relative error due to its scale-independence [7].
Chapter 3

Numerical Stability of the IPDIP Algorithm

In this chapter, we will discuss the numerical stability of the IPDIP implementation. Specifically, we will cover the computational steps involved in solving the linear-quadratic system of equations and discuss the stability issues involved in solving it.

3.1 Solving the Linear System

In computing a predictor step in the IPDIP method for Algorithm 2, the right-hand-side (RHS) of Equation (2.11), $-F(q^k)$, can be defined to be the following vector of residuals:

$$F(q^k) = \begin{bmatrix} r_p^k \\ r_u^k \\ r_d^k \\ r_{xz}^k \\ r_{sw}^k \end{bmatrix} = \begin{bmatrix} Ax^k - b \\ x_u^k + s^k - u \\ A^T y^k + z^k - \text{app}(w^k) - c \\ x^k \circ z^k \\ s^k \circ w^k \end{bmatrix}, \quad (3.1)$$

where $r_p^k$, $r_u^k$, and $r_d^k$ are, respectively, the primal, upper bound, and dual residuals, while $r_{xz}^k$ and $r_{sw}^k$ are the complementarity residuals.
The corrector step, on the other hand, has the following right-hand-side (RHS) vector of residuals:

\[
F(q^k) = \begin{bmatrix}
r_p^k \\
r_u^k \\
r_d^k \\
r_{xz}^k \\
r_{sw}^k
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
\Delta x_{\text{aff}}^k & \Delta z_{\text{aff}}^k - \mu^k e \\
\Delta s_{\text{aff}}^k & \Delta w_{\text{aff}}^k - \mu^k e
\end{bmatrix},
\]

(3.2)

where \( e \) is a vector of ones with the proper dimensions.

In order to execute the steps in Algorithm 2, we take advantage of the sparsity of \( F'(q) \) to solve the following system of equations:

\[
A\Delta x = -r_p 
\]

(3.3)

\[
\hat{I}^T \Delta x + I_u \Delta s = -r_u 
\]

(3.4)

\[
A^T \Delta y - \hat{I} \Delta w + I \Delta z = -r_d 
\]

(3.5)

\[
Z\Delta x + X\Delta z = -r_{xz} 
\]

(3.6)

\[
W\Delta s + S\Delta w = -r_{sw} 
\]

(3.7)

By block Gaussian elimination:

\[
(ADA^T)\Delta y = -(r_p + AD\hat{r}_d) 
\]

(3.8)

\[
\Delta x = DA^T(\Delta y + r_d) 
\]

(3.9)

\[
\Delta z = -(X)^{-1}(Z\Delta x + r_{xz}) 
\]

(3.10)

\[
\Delta s = -(\Delta x_u + r_u) 
\]

(3.11)

\[
\Delta w = -(S)^{-1}(W\Delta s + r_{sw}), 
\]

(3.12)

where

\[
D = (X^{-1}Z + \text{diag(app}([S]^{-1}w)))^{-1} 
\]

(3.13)

and

\[
\hat{r}_d = r_d - X^{-1}r_{xz} + \text{app}(S^{-1}(r_{sw} - Wr_u)) 
\]

(3.14)
Solving for $\Delta y$ is the most difficult computation in the system of equations. The rest can be simply solved by back-substitution. Note that $D$ is a positive diagonal matrix, and $ADA^T$ is a symmetric positive semi-definite coefficient matrix. The matrix is positive definite if $A$ is full rank. If $A$ is full rank, $ADA^T$ can be factorized into $ADA^T = R^T R$ via a sparse Cholesky decomposition where $R$ is upper triangular. Thus, two backsolves can be used to solve for $\Delta y$ in $R^T R \Delta y = -(r_p + AD\hat{r}_d)$. In order to ensure the sparsity of $R$, we can reorder the rows and columns of the sparse $A$ by minimum-degree ordering, nested dissection, or minimum local fill.

### 3.2 Ill-Conditioning

Sparse Cholesky factorization is stable, but it guarantees accurate solutions only for well-conditioned problems. During the aforementioned reordering phase, the row and column exchanges of $A$ are reordered based upon the sparsity pattern of $R$ and not on the resulting numerical values of the nonzero elements. Thus, the coefficient matrix, $ADA^T$, may become highly ill-conditioned as iterates near the solution.

For every solution $(x^*, s^*, y^*, w^*, z^*)$, the complementarity conditions require the following to occur:

\[
\begin{cases}
  x^* = 0 \text{ and/or } s^* = 0 \\
  w^* = 0 \text{ and/or } z^* = 0
\end{cases}
\]

Let us create the following two index sets:

\[
\begin{align*}
  \mathcal{B} &= \{i | x_i^* > 0 \text{ for all } i \text{ and } 0 < x_i^* < u_i \text{ for } i \leq n_a \} \\
  \mathcal{M} &= \{i | x_i^* = 0 \text{ or } x_i^* = u_i \text{ for } i \leq n_a \}
\end{align*}
\]

Note that $\mathcal{B}$ and $\mathcal{M}$ are disjoint sets and that the diagonal values in $D$ will either asymptotically approach 0 or $\infty$ as the non-negativity variables approach the solution:

\[
D_{ii} \rightarrow \begin{cases} \infty, & i \in \mathcal{B} \\ 0, & i \in \mathcal{M} \end{cases}
\]
Using $\mathcal{B}$ and $\mathcal{M}$ to partition Equation (3.8), we obtain:

$$
(A_B D_B A_B^T + A_N D_N A_N^T) \Delta y = -(r_p + A_B D_B(\hat{r}_d)_B + A_N D_N(\hat{r}_d)_N),
$$

where $D = \text{diag}(D_B, D_N)$, $A = [A_B, A_N]$, and $\hat{r}_d = ((\hat{r}_d)_B, (\hat{r}_d)_N)$.

As the iterates converge to the solution, Equation (3.16) reduces to approximately:

$$
(A_B D_B A_B^T) \Delta y = -(r_p + A_B D_B(\hat{r}_d)_B),
$$

(3.17)

since $A_N D_N A_N^T \to 0$ and $A_N D_N(\hat{r}_d)_N \to 0$. If the iterates converge to the solution, $r_p$ should also go to zero.

Note that $(A_B D_B A_B^T) \to \infty$ and $-(A_B D_B(\hat{r}_d)_B) \to \infty$ in Equation (3.17) as the iterates near the solution. There are instances when all eigenvalues of the matrix $ADA^T$ grow to $\infty$ at the same rate, and so $ADA^T$ remains well-conditioned. However, in most cases, the condition number of $ADA^T$ may dramatically increase during the latter iterations [17]. This can be due to many reasons. In fact, whenever LP is primal degenerate ($|\mathcal{B}| < m$), $ADA^T$ necessarily approaches a singular matrix. Thus, the coefficient matrix, $ADA^T$, may become increasingly ill-conditioned towards the end of the iterations. This can result in significant numerical errors in the computed solution and leads us to wonder whether the ill-conditioning is playing a significant role in lowering the solution accuracy of a problem.

The numerical stability issues of IPDIP algorithms have been studied by a few researchers, most notably by Stephen Wright [16], [18], [19]. The most relevant work to this dissertation is [19], where the author studied the same algorithmic framework as the one described in Chapter 2. The main conclusion of [19] is that, under suitable assumptions on the data, the best accuracy that can be generally expected of the IPDIP algorithm (with the block Gaussian elimination procedure) is on the order of $\sqrt{\epsilon}$, where $\epsilon$ is the machine epsilon. That is, in the usual double precision 64-bit floating point storage scheme, the expected accuracy is in general on the order of $10^{-8}$. Wright derives this conclusion by studying how the roundoff errors introduced during the modified Cholesky factorization, the errors from the triangular substitutions, and
the errors from the matrix factored contributed to the quality of the search direction. He discovers that during the late-stage iterations of the IPDIP algorithm, when the duality gap decreases below $\sqrt{\epsilon}$, the effect of the errors on the step length and the computed residuals become acute, thereby limiting the solution accuracy of the IPDIP algorithm.

In [19], Wright conducted some numerical experiments to validate his theory on some small test problems. At this point, we are not aware of more comprehensive numerical studies conducted on a large set of relatively large problems.
Chapter 4

Solution Accuracy of the IPDIP Algorithm

In this chapter, we will investigate the solution accuracy of the IPDIP algorithm from a numerical standpoint. We use the linear programming solver, LIPSOL, and a set of Netlib test problems as the testing environment. Running LIPSOL on the Netlib problems with varying convergence criterions, we observe the solution accuracy of the IPDIP algorithm.

4.1 Test Environment - LIPSOL

There are many software packages available for solving linear programming problems via interior-point methods, such as PCx [5], which is written in Fortran and C++, BPMPD [11], which is written in Fortran, and APOS [1], which is written in C++. Because of the convenience of Matlab, we choose to work in LIPSOL.

LIPSOL (Linear programming Interior Point SOLvers) [22] is a Matlab-based software package for solving linear programming problems via an infeasible primal-dual interior-point (IPDIP) algorithm. It can handle fairly many mid-scale linear programming problems because it exploits Matlab’s sparse-matrix functions and ex-
ternal interface facilities. The algorithmic framework discussed in Chapter 2 is based on LIPSOL's implementation of interior-point methods. In this section, we mention a few more intricacies specific to LIPSOL and relevant in later discussions.

LIPSOL utilizes a modified version of the sparse Cholesky code written by Ng and Peyton [13]. It handles problematic pivot elements by replacing them with workable values. If a diagonal pivot is very close to zero, the corresponding element of its Cholesky factor is replaced by a huge number.

In LIPSOL, the centering parameter, $\sigma^k$, has a fixed upper bound of 0.208 for all iterations. During the end-stage iterations, indicated by the total relative error falling below $10^{-2}$ and the duality gap becoming less than $10^{-3}$, $\sigma^k$ is forced to have the same order as the total relative error. This adaptive strategy helps induce faster local convergence according to [23] and [20].

In determining the step length parameters, $\alpha_p$ and $\alpha_d$, LIPSOL requires that a full step ($\alpha_p = 1$ and/or $\alpha_d = 1$) in the primal and/or dual direction(s) be taken if the new iterate lies within the relative interior of the nonnegative orthant. On the other hand, if a full step is not possible, $\alpha_p$ and $\alpha_d$ are successively decreased until the new iterate falls in the set of $\mathcal{N}$, a neighborhood of the central path. Thus, membership of the neighborhood [12] is not strictly enforced.

We use LIPSOL as a testing environment to study the solution accuracy of the IPDIP algorithm. The default tolerance for accuracy in LIPSOL is set to the standardly accepted $10^{-8}$, which is also the default tolerance level at which PCx declares optimality [5].

4.2 Test Problems - Netlib

In order to study solution accuracy from an empirical standpoint, we use a set of 95 Netlib [6] test problems. Netlib contains a collection of linear programs from a variety of sources, including real applications, which has become a standard test problem set
used for testing linear programming algorithms and software. The problems range in size from 32 to 15,695 variables. Furthermore, they are all feasible with optimal solutions.

Some problems can be grouped into “families” due to similarities in the way they were constructed. For example, greenbea and greenbeb are the same problems with different bounds. The pairs, fit1p/fit1d and fit2p/fit2d, are primal and dual versions of the same two problems, with the exception that the objective functions of the dual versions have been negated in order to convert them into minimization problems. This information becomes useful when we observe how similar problems behave under different environments.

4.3 Solution Accuracy Under Different Tolerances

In Tables A.1 to A.4, we present some results for the 95 Netlib problems solved by LIPSOL under the $10^{-8}$ default tolerance setting. From the table, we note that out of the 95 test problems, only two problems, df1001 and greenbea, failed to achieve high enough accuracy to satisfy the convergence criterion. Furthermore, as we successively decreased the convergence criterion in steps of two orders of magnitude from $10^{-8}$ to $10^{-16}$, we observe in Table 4.1 that more and more problems fail to achieve high enough accuracy to meet the criterion. Note that LIPSOL is able to attain the accuracy of $10^{-14}$ for over half of the problems. This is somewhat surprising to us since Wright [19] concluded from his error analysis that $10^{-8}$ is the expected accuracy from IPDIP algorithms. Our results do not disprove Wright’s conclusions, but suggests that perhaps other interactions may work into play allowing the IPDIP algorithm to attain accuracy higher than what Wright’s analysis concludes.

Note that slow convergence to the solution is not the reason for some of the test problems failing to meet the tolerance criteria. The maximum number of iterations allowed is 100, so problems failing to meet the tolerance criteria prior to 100 iterations
suffered a deterioration in their primal, dual, or gap residuals during latter iterations. For problems that reached 100 iterations without satisfying the tolerance criteria, their primal, dual, or gap residuals eventually reached their potential minimums and remained constant for the remaining iterations without meeting the tolerance criteria.

<table>
<thead>
<tr>
<th>Tolerance</th>
<th># of Problems</th>
<th>% of Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-8}$</td>
<td>93</td>
<td>98%</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>89</td>
<td>94%</td>
</tr>
<tr>
<td>$10^{-12}$</td>
<td>84</td>
<td>88%</td>
</tr>
<tr>
<td>$10^{-14}$</td>
<td>54</td>
<td>57%</td>
</tr>
<tr>
<td>$10^{-16}$</td>
<td>6</td>
<td>6%</td>
</tr>
</tbody>
</table>

Table 4.1: The number of problems, out of 95 Netlib problems, for which LIPSOL achieves high enough accuracy to meet the convergence criterion.

Evidently, there exists a wide range of difficulty levels in solving problems in the test set. We can construct two groups of problems: 1) an "easy" group, for the problems that are able to satisfy strict tolerance criteria; 2) a "hard" group, for problems that are unable to meet low tolerance criteria. The "hard" problems are the nine problems failing to meet a convergence criterion of $10^{-12}$ or higher. They include df1001, greenbea, seba, grow7, grow15, grow22, fit1p, fit2p, and pilot87. We have two sets of "easy" problems. One set is selected based on accuracy alone, while the other one takes the size of the problem into consideration. Our purpose for the latter case is to be able to select easy problems that may take more effort to solve due to their large size. The "easy" set based on accuracy alone are the ten problems achieving the highest accuracy when we solve with LIPSOL at the convergence criterion of $10^{-16}$: sc50b, afiro, shell, israel, sierra, sc50a, adlittle, lotfi, standata, and maros-r7. The "easy" group of problems, taking size into consideration, include: maros-r7, pilot, d6cube, nesm, ship08l, pilotnov, sctap3, sierra, 24fv47, and ship08s.
4.4 A Look at Condition Numbers

The condition number of a matrix measures the sensitivity of a matrix to perturbations. A well-conditioned matrix has a relatively small condition number, while an ill-conditioned matrix has a relatively large condition number. We use the 1-norm condition number estimate in Matlab to measure the condition numbers for the coefficient matrix of (3.8) at each iteration of a problem.

As discussed in the previous section, we can construct two sets of problems for purposes of comparison. Tables 4.2 and 4.3 list, respectively, the ten “easy” and eight “hard” problems, which have been selected based upon their size and accuracy. Normally, one would expect the condition numbers for the “easy” problems to be lower than that of the “hard” problems, since “easy” problems are able to satisfy convergence criteria that hard problems do not to meet. On the contrary, it can be seen that no such trend exists. For most problems, the condition numbers for both “easy” and “hard” problems start at low values of similar magnitudes and increase to high values of similar magnitudes. Each set also contains problems whereby the condition numbers do not grow large relative to other problems of similar size and accuracy.

The graphs in Figures 4.1 to 4.5 show the plot of the estimated condition numbers per iteration for each problem studied. With the exception of a few problems, both “easy” and “hard” problems experience significant increases in condition number with iteration increases. A few problems have relatively small condition numbers regardless of being categorized as “easy” or “hard” problems.

We conclude from our observations that condition numbers cannot be used as the sole indicator of how accurate the final solution is. Error from a source other than the ill-conditioned coefficient matrix of (3.8) must also be contributing to lowering the overall accuracy of problem solutions. The primary suspects are errors in computing

\footnote{Due to its highly ill-conditioned nature, a condition number estimate could not be obtained for \texttt{df1001}, so the problem was not included as one of the “hard” problems in this study.}
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Bytes</th>
<th>TRE</th>
<th>Cond0</th>
<th>Cond1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>maros-r7</td>
<td>4,812,587</td>
<td>7.12.e-16</td>
<td>1.36.e+01</td>
<td>2.02.e+05</td>
</tr>
<tr>
<td>2</td>
<td>pilot</td>
<td>278,593</td>
<td>2.17.e-15</td>
<td>1.19.e+05</td>
<td>6.31.e+31</td>
</tr>
<tr>
<td>3</td>
<td>d6cube</td>
<td>167,633</td>
<td>3.25.e-15</td>
<td>7.74.e+06</td>
<td>1.76.e+46</td>
</tr>
<tr>
<td>4</td>
<td>nesm</td>
<td>117,828</td>
<td>1.10.e-15</td>
<td>1.23.e+03</td>
<td>2.73.e+13</td>
</tr>
<tr>
<td>5</td>
<td>ship08l</td>
<td>117,083</td>
<td>1.63.e-15</td>
<td>1.09.e+05</td>
<td>5.80.e+39</td>
</tr>
<tr>
<td>6</td>
<td>pilotno</td>
<td>89,779</td>
<td>1.43.e-15</td>
<td>3.57.e+05</td>
<td>9.23.e+31</td>
</tr>
<tr>
<td>7</td>
<td>sctap3</td>
<td>78,688</td>
<td>5.89.e-16</td>
<td>1.19.e+05</td>
<td>2.78.e+33</td>
</tr>
<tr>
<td>8</td>
<td>sierra</td>
<td>76,627</td>
<td>2.29.e-15</td>
<td>9.91.e+17</td>
<td>8.10.e+33</td>
</tr>
<tr>
<td>9</td>
<td>24fv47</td>
<td>70,477</td>
<td>9.06.e-16</td>
<td>8.46.e+04</td>
<td>Inf</td>
</tr>
<tr>
<td>10</td>
<td>ship08s</td>
<td>70,093</td>
<td>1.23.e-15</td>
<td>3.56.e+04</td>
<td>9.57.e+34</td>
</tr>
</tbody>
</table>

Table 4.2: Starting (Cond0) and ending (Cond1) condition numbers for “easy” problems. “Easy” problems are selected based on the ten largest problems with TRE \( \leq 5 \times 10^{-15} \) when the convergence criterion is \( 10^{-14} \). TRE is the total relative error, and in this case, represents the solution accuracy at which LIPSOL solves the problem.

the right-hand-side.

4.5 Primal Residual

We look more closely at the residuals for each of the “hard” problems\(^2\) in order to determine which component of the total relative error is responsible for affecting the overall lower accuracy. The plots illustrated in Figures 4.6 to 4.8 show the general trend of the primal, dual, and gap residuals for the “hard” problems. It becomes evident that in most cases, the primal residual plays a determining role in signifying whether a problem will deteriorate in accuracy. Furthermore, the total relative error appears to closely follow the trend of the primal residual, especially during the latter iterations.

\(^2\)Including df1001
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Bytes</th>
<th>TRE</th>
<th>Cond0</th>
<th>Cond1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pilot87</td>
<td>514,192</td>
<td>2.09.e-11</td>
<td>1.29.e+05</td>
<td>3.42.e+32</td>
</tr>
<tr>
<td>2</td>
<td>fit2p</td>
<td>439,794</td>
<td>3.96.e-07</td>
<td>2.56.e+03</td>
<td>8.88.e+28</td>
</tr>
<tr>
<td>3</td>
<td>greenbea</td>
<td>235,711</td>
<td>3.77.e-07</td>
<td>2.61.e+05</td>
<td>3.73.e+31</td>
</tr>
<tr>
<td>4</td>
<td>fit1p</td>
<td>65,116</td>
<td>1.14.e-07</td>
<td>1.78.e+03</td>
<td>3.66.e+29</td>
</tr>
<tr>
<td>5</td>
<td>grow22</td>
<td>50,789</td>
<td>8.50.e-10</td>
<td>4.31.e+01</td>
<td>2.13.e+04</td>
</tr>
<tr>
<td>6</td>
<td>seba</td>
<td>38,627</td>
<td>1.07.e-06</td>
<td>5.13.e+05</td>
<td>2.21.e+36</td>
</tr>
<tr>
<td>7</td>
<td>grow15</td>
<td>35,041</td>
<td>4.88.e-10</td>
<td>4.30.e+01</td>
<td>1.18.e+05</td>
</tr>
<tr>
<td>8</td>
<td>grow7</td>
<td>17,043</td>
<td>3.19.e-10</td>
<td>4.07.e+01</td>
<td>2.15.e+04</td>
</tr>
</tbody>
</table>

Table 4.3: Starting (Cond0) and ending (Cond1) condition numbers for “hard” problems. “Hard” problems are selected based on the nine largest problems with TRE $\geq 10^{-12}$ when the convergence criterion is $10^{-14}$.

In order to observe how the primal residual of an “easy” problem will behave compared to the primal residual of a “hard” problem, we study the difference between the residual behavior for two very similar problems: fit2p, which fails the convergence criterion of $10^{-12}$, and its negated dual problem, fit2d, which meets the convergence criterion of $10^{-12}$. From Figure 4.9, it can be observed that the primal residual in fit2d converges faster than that of fit2p. Furthermore, the drop in the primal residual for fit2d occurs at iteration five while the drop in the primal residual for fit2p is spread out over the course of ten iterations. The total relative error for both eventually decreases to approximately $10^{-11}$. However, only fit2p is in the condition to improve past this point. From our observations, it appears that the “easy” problem satisfies primal feasibility early while the “hard” problem satisfies primal feasibility gradually. We thus conjecture that it is more beneficial for the primal residual to quickly converge to high accuracy during earlier, as opposed to later, iterations. In Section 4.4, we concluded that the condition number of the coefficient matrix for Equation (3.8) is not solely responsible for the ill-conditioning
of a problem and suspect the right-hand-side of the equation as a major contributor to errors. Looking more closely at the right-hand side, \(-(r_p + AD\hat{r}_d)\), of Equation (3.8), we believe that the primal residual term \(r_p\) in the right-hand side could be a pivotal player in the process of generating inaccurate solutions.

Recalling Equation (3.17), as the iterates approach a boundary, \(A_B D_B A_B^T\) and \(A_B D_B (\hat{r}_d)_B\) grow towards \(\infty\); hence the matrix \(ADA^T\) is dominated by \(A_B D_B A_B^T\). In many, if not most, cases, the matrix \(A_B D_B A_B^T\) is rank-deficient, leading to the ill-conditioning of \(ADA^T\). However, the equation

\[(A_B D_B A_B^T)\Delta y = -A_B D_B (\hat{r}_d)_B\]

is always consistent and still capable of producing a good step, since the right-hand side resides in the column space of the coefficient matrix. On the other hand, the
term \( r_p \) is generally not restricted to the same subspace. If the size of \( r_p \) has not been sufficiently reduced when \( ADA^T \) becomes severely ill-conditioned, then we are facing the approximate equation

\[
(ABA_B^T)\Delta y = -(r_p + A_B D_B (\tilde{r}_d)_B),
\]

which is not consistent and can give bad steps. Namely, if the primal residual does not converge fast enough during earlier iterations, then its presence in the right-hand-side of a highly ill-conditioned equation will cause numerical errors. On the other hand, if the primal residual quickly converges to high accuracy during earlier iterations, then the presence of \( r_p \) becomes numerically insignificant during later iterations. This explains well why in our numerical experiments, problems with early convergence of primal residual have usually achieved high accuracy at the end.
Figure 4.3: Condition Numbers for "Easy" Problems

In order to see if our conjecture can be applied to problems other than the fit2 problems, we plot in Figure 4.10 the magnitude of biggest order primal residual decrease at an iteration of a problem for ten “easy” problems versus nine “hard” problems\(^3\). We note that “easy” problems tend to have larger magnitudes of order decrease at earlier iterations while most “hard” problems have relatively smaller decrease over a wider iteration range.

Interestingly, the condition numbers for fit2d and fit2p start at similar low values of \(3.15 \times 10^3\) and \(2.56 \times 10^3\), respectively, but the condition number for fit2d only grows to \(2.79 \times 10^5\) while the condition number for fit2p rapidly grows to \(8.88 \times 10^28\). In this case, the coefficient matrix, \(D\), may also be contributing to the poor accuracy of fit2p.

4.6 Data Truncation

Recalling the three “hard” problems which have low condition numbers (the grow problems), we realized that they are also the three exception problems in Figure 4.10 that experience a sharp decrease in primal residual at earlier iterations despite their ultimate low accuracy. This leads us to question whether data truncation may have

\(^3\)Including df1001
played a role in their inability to attain high accuracy. Recall that when a problem takes a full step length of 1 in the primal direction, without numerical errors, it would completely correct for the primal residual error in one step. Thus, the relative primal error can be written as:

$$\min \frac{\|Ax - b\|}{\|b\|}$$ (4.1)

where the accuracy should be the minimal attainable. For the grow problems, full steps in the primal direction results in a relative primal error of no less than $\approx 10^{-10}$, which is approximately the minimum attainable by its total relative error. Since the grow problems have relatively low condition numbers for all iterations, and a corrected primal step of length 1 cannot go beyond a tolerance of $10^{-10}$, we highly suspect that the problem lies behind a truncated dataset. When we remove the problems from Figure 4.5, our plot perfectly illustrates the trend.
Figure 4.5: Condition Numbers for "Hard" Problems

Data truncation may be playing a role in lowering the accuracy of the grow problems because they have low condition numbers despite their classification as "hard" problems. This does not mean that the accuracy of other problems with high condition numbers are not affected by data truncation. To determine this, more time will be needed for investigation.
Figure 4.6: Residual Behavior for df1001.
Figure 4.7: Residual Behavior for seba
Figure 4.8: Residual Behavior for **grow7**
Figure 4.9: Residual Behavior for fit2 Problems
Figure 4.10: Biggest Order of Primal Residual Decrease for All Iterations. This plot includes ten "easy" problems and nine "hard" problems. The "easy" problems are the ten problems achieving the highest accuracy when LIPSOL's convergence criterion is set at $10^{-16}$. Under such conditions, the problems have $\text{TRE} \leq 2.\text{e-16}$. The problems include sc50b, afiro, shell, israel, sierra, sc50a, adlittle, lotfi, standata, and maros-r7. The "hard" problems include a collection of nine problems with the lowest accuracy when we run LIPSOL with a convergence criterion set at $10^{-12}$ or higher, depending on which convergence criterion the problem first fails to meet. The problems include df1001, greenbea, seba, grow7, grow15, grow22, fit1p, fit2p, and pilot87.
Chapter 5

Conclusions and Future Work

In this chapter, we state the main conclusions from our work, briefly describe some experiments we had performed in an attempt to reduce the solution accuracy of the IPDIP algorithm, and discuss future experiments that can be performed.

5.1 Conclusions

From our numerical study, we make the following remarks:

- IPDIP, as implemented in LIPSOL, can solve most Netlib problems to a much higher accuracy than the standard $10^{-8}$ accuracy.
- Condition number is not the sole factor responsible for lowering solution accuracy in problems.
- Primal residual is the residual component causing the rise in total relative error in most problems.
- Early satisfaction of the primal equality constraints is often conducive to achieving high solution accuracy.
5.2 Some Experiments: Room for Future Work

We performed several experiments in an attempt to reduce the solution accuracy of the IPDIP algorithm and discovered that the problem was harder than we had initially expected. In this section, we briefly discuss a few of the many experiments we had attempted and the obstacles we encountered. We did not have sufficient time to implement some of our experimental ideas, but we discuss future work that can be done.

5.2.1 Early Satisfaction of Primal Feasibility

In accordance with our discussion in Section 4.1, we observed that whenever a sharp decrease in the primal residual occurs, a full step is taken in the primal direction ($\alpha_p = 1$) \(^1\). Thus, from our conjecture in Section 4.5, taking larger steps can potentially alleviate ill-conditioning if they are taken early. We tried to facilitate faster convergence by centralizing the path during initial iterations. By setting $\sigma^k = 1$ for the first 3 iterations, we tried to get closer to the central path, allowing for the possibility of bigger steps to be taken early on in the problem. Overall, our experimental group performed worse than the control group when the tolerance for accuracy is lowered to $10^{-12}$ or lower, with the exception of only two “hard problems”. We propose that for future experiments, pure centering steps should be taken in the beginning iterations.

Similar to what we attempted in the previous experiment, a future experiment we can attempt is to alter parameters in order to create an environment that would be more conducive to an early satisfaction of the primal equality constraint.

\(^1\)We have confirmed this with numerical experiments.
5.2.2 Reduce the RHS Error

When we discovered that the coefficient matrix is not solely responsible for solution accuracy, we concluded that the RHS must also play a factor in the ill-conditioning. RHS represents the right-hand-side of a linear system, and in this case, is defined as:

\[ RHS = -(r_p + AD\hat{r}_d) \] (5.1)

From our data, we had observed that the primal residual is the component causing the rise in total relative error. Thus, we suspected that lower accuracy may be partly attributed to errors in \( r_p \). For some problems, data truncation may be playing a role in preventing a problem from obtaining a small \( r_p \). For other problems that don’t have truncated data, there may exist a better way to compute \( r_p \) which would reduce its computational errors. So far, we have been unsuccessful in finding a better way to compute the RHS without introducing new errors into the problem. However, this will be an interesting problem for future investigation.

5.2.3 Other Experiments

We have tried experiments where a small step is taken back whenever the duality gap is small and the primal residual is large. These back-tracking steps are usually taken during the latter iterations, and its purpose is to step out of a highly ill-conditioned region near the solution. However, we concluded that when the primal residual is large and the iterate is near the solution, the problem has already become too highly sensitive and, thus, too unstable to improve the solution.

We have also concluded that the IPDIP algorithm is highly sensitive to initial values. During the testing phase of our experiments, we allow only the first iteration to be affected by changes. Surprisingly, a small change during the beginning iterations, even if leading to an increase in residual error when compared to the control group, may lead to improvements for some “hard” problems.
5.3 Final Remarks

We discovered that improving the solution accuracy beyond the normally accepted accuracy of $10^{-8}$ is a difficult endeavor. For most of the problems, the Chur complement matrices of the form $ADA^T$ become highly ill-conditioned. We’ve discovered that we can improve individual problems just by making small alterations to the code, but have been unable to find a method which will result in a general improvement for most problems. However, from our numerical study, we have been able to better understand why $r_p$ may be a highly suspect source of error affecting solution accuracy, which may be able to help future investigators in this area.
Appendix A

LIPSOL Results from Solving Netlib Problems
<table>
<thead>
<tr>
<th>Problem</th>
<th>Converged</th>
<th>Iter</th>
<th>TRE</th>
<th>Obj. Value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>25fv47</td>
<td>1</td>
<td>25</td>
<td>1.09e-14</td>
<td>5.5018458883e+03</td>
<td>4.72</td>
</tr>
<tr>
<td>80bau3b</td>
<td>1</td>
<td>39</td>
<td>8.49e-09</td>
<td>9.8722419246e+05</td>
<td>20.15</td>
</tr>
<tr>
<td>adlittle</td>
<td>1</td>
<td>13</td>
<td>5.85e-13</td>
<td>2.2549496316e+05</td>
<td>0.31</td>
</tr>
<tr>
<td>afiro</td>
<td>1</td>
<td>8</td>
<td>1.97e-13</td>
<td>-4.6475314286e+02</td>
<td>0.19</td>
</tr>
<tr>
<td>agg</td>
<td>1</td>
<td>21</td>
<td>1.06e-11</td>
<td>-3.5991767287e+07</td>
<td>1.88</td>
</tr>
<tr>
<td>agg2</td>
<td>1</td>
<td>18</td>
<td>4.30e-11</td>
<td>-2.0239252356e+07</td>
<td>2.11</td>
</tr>
<tr>
<td>agg3</td>
<td>1</td>
<td>17</td>
<td>6.46e-11</td>
<td>1.0312115935e+07</td>
<td>2.01</td>
</tr>
<tr>
<td>bandm</td>
<td>1</td>
<td>18</td>
<td>8.55e-11</td>
<td>-1.5862801844e+02</td>
<td>0.87</td>
</tr>
<tr>
<td>beaconfd</td>
<td>1</td>
<td>13</td>
<td>2.79e-11</td>
<td>3.3592485807e+04</td>
<td>0.71</td>
</tr>
<tr>
<td>blend</td>
<td>1</td>
<td>12</td>
<td>7.81e-14</td>
<td>-3.0812149846e+01</td>
<td>0.32</td>
</tr>
<tr>
<td>bnl1</td>
<td>1</td>
<td>26</td>
<td>2.95e-11</td>
<td>1.9776295615e+03</td>
<td>2.28</td>
</tr>
<tr>
<td>bnl2</td>
<td>1</td>
<td>31</td>
<td>4.78e-09</td>
<td>1.8112365463e+03</td>
<td>9.49</td>
</tr>
<tr>
<td>boeing1</td>
<td>1</td>
<td>21</td>
<td>3.44e-10</td>
<td>-3.3521356739e+02</td>
<td>1.86</td>
</tr>
<tr>
<td>boeing2</td>
<td>1</td>
<td>19</td>
<td>5.67e-10</td>
<td>-3.1501872791e+02</td>
<td>0.89</td>
</tr>
<tr>
<td>bore3d</td>
<td>1</td>
<td>18</td>
<td>1.09e-11</td>
<td>1.3730803942e+03</td>
<td>0.82</td>
</tr>
<tr>
<td>brandy</td>
<td>1</td>
<td>17</td>
<td>1.59e-13</td>
<td>1.5185098965e+03</td>
<td>0.73</td>
</tr>
<tr>
<td>capri</td>
<td>1</td>
<td>20</td>
<td>5.77e-14</td>
<td>2.6900129138e+03</td>
<td>1.21</td>
</tr>
<tr>
<td>cycle</td>
<td>1</td>
<td>25</td>
<td>1.38e-09</td>
<td>-5.2263924013e+00</td>
<td>10.65</td>
</tr>
<tr>
<td>czprob</td>
<td>1</td>
<td>36</td>
<td>5.88e-09</td>
<td>2.1851966989e+06</td>
<td>4.48</td>
</tr>
<tr>
<td>d2q06c</td>
<td>1</td>
<td>32</td>
<td>6.52e-10</td>
<td>1.2278421082e+05</td>
<td>21.59</td>
</tr>
<tr>
<td>d6cube</td>
<td>1</td>
<td>23</td>
<td>4.71e-11</td>
<td>3.1549166693e+02</td>
<td>9.22</td>
</tr>
<tr>
<td>degen2</td>
<td>1</td>
<td>14</td>
<td>1.89e-10</td>
<td>-1.4351779999e+03</td>
<td>1.33</td>
</tr>
<tr>
<td>degen3</td>
<td>1</td>
<td>20</td>
<td>5.08e-09</td>
<td>-9.8729399275e+02</td>
<td>14.15</td>
</tr>
<tr>
<td>df001</td>
<td>0</td>
<td>71</td>
<td>1.23e-07</td>
<td>1.1266395658e+07</td>
<td>3021.26</td>
</tr>
<tr>
<td>e226</td>
<td>1</td>
<td>21</td>
<td>1.12e-13</td>
<td>-1.8751929066e+01</td>
<td>1.07</td>
</tr>
</tbody>
</table>

Table A.1: Set of 95 Netlib problems solved under the default tolerance setting of $10^{-8}$
<table>
<thead>
<tr>
<th>Problem</th>
<th>Converged</th>
<th>Iter</th>
<th>TRE</th>
<th>Obj. Value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>etamacro</td>
<td>1</td>
<td>25</td>
<td>9.18e-09</td>
<td>-7.5571523268e+02</td>
<td>1.59</td>
</tr>
<tr>
<td>fffff800</td>
<td>1</td>
<td>26</td>
<td>5.76e-09</td>
<td>5.5567956482e+05</td>
<td>2.94</td>
</tr>
<tr>
<td>finnis</td>
<td>1</td>
<td>30</td>
<td>3.92e-12</td>
<td>1.7279106560e+05</td>
<td>2.03</td>
</tr>
<tr>
<td>fit1d</td>
<td>1</td>
<td>19</td>
<td>3.04e-11</td>
<td>-9.1463780924e+03</td>
<td>2.90</td>
</tr>
<tr>
<td>fit1p</td>
<td>1</td>
<td>16</td>
<td>4.54e-10</td>
<td>9.1463780922e+03</td>
<td>13.21</td>
</tr>
<tr>
<td>fit2d</td>
<td>1</td>
<td>22</td>
<td>5.90e-10</td>
<td>-6.8464293283e+04</td>
<td>31.14</td>
</tr>
<tr>
<td>fit2p</td>
<td>1</td>
<td>21</td>
<td>4.49e-11</td>
<td>6.8464293294e+04</td>
<td>42.70</td>
</tr>
<tr>
<td>forplan</td>
<td>1</td>
<td>22</td>
<td>2.78e-12</td>
<td>-6.6421896098e+02</td>
<td>1.46</td>
</tr>
<tr>
<td>ganges</td>
<td>1</td>
<td>18</td>
<td>1.89e-10</td>
<td>-1.0958573613e+05</td>
<td>2.60</td>
</tr>
<tr>
<td>giffpin</td>
<td>1</td>
<td>21</td>
<td>4.42e-13</td>
<td>6.9022359995e+06</td>
<td>1.41</td>
</tr>
<tr>
<td>greenbea</td>
<td>0</td>
<td>37</td>
<td>3.77e-07</td>
<td>-7.2524393528e+07</td>
<td>17.67</td>
</tr>
<tr>
<td>greenbeb</td>
<td>1</td>
<td>38</td>
<td>6.85e-13</td>
<td>-4.3022602612e+06</td>
<td>19.68</td>
</tr>
<tr>
<td>grow15</td>
<td>1</td>
<td>17</td>
<td>7.90e-10</td>
<td>-1.0687094129e+08</td>
<td>1.95</td>
</tr>
<tr>
<td>grow22</td>
<td>1</td>
<td>19</td>
<td>8.19e-10</td>
<td>-1.6083433648e+08</td>
<td>2.98</td>
</tr>
<tr>
<td>grow7</td>
<td>1</td>
<td>16</td>
<td>4.84e-10</td>
<td>-4.7787811815e+07</td>
<td>1.08</td>
</tr>
<tr>
<td>israel</td>
<td>1</td>
<td>23</td>
<td>7.81e-09</td>
<td>-8.9664482186e+05</td>
<td>1.94</td>
</tr>
<tr>
<td>kb2</td>
<td>1</td>
<td>15</td>
<td>1.08e-11</td>
<td>-1.7499001299e+03</td>
<td>0.46</td>
</tr>
<tr>
<td>lotfi</td>
<td>1</td>
<td>18</td>
<td>3.05e-13</td>
<td>-2.5264706062e+01</td>
<td>0.58</td>
</tr>
<tr>
<td>maros</td>
<td>1</td>
<td>33</td>
<td>1.11e-12</td>
<td>-5.8063743701e+04</td>
<td>5.43</td>
</tr>
<tr>
<td>maros-r7</td>
<td>1</td>
<td>15</td>
<td>7.59e-09</td>
<td>1.4971851665e+06</td>
<td>120.15</td>
</tr>
<tr>
<td>modszk1</td>
<td>1</td>
<td>24</td>
<td>6.73e-15</td>
<td>3.2061972899e+02</td>
<td>1.95</td>
</tr>
<tr>
<td>nesm</td>
<td>1</td>
<td>33</td>
<td>5.38e-09</td>
<td>1.4076036488e+07</td>
<td>7.39</td>
</tr>
<tr>
<td>perold</td>
<td>1</td>
<td>31</td>
<td>5.97e-11</td>
<td>-9.3807552782e+03</td>
<td>4.08</td>
</tr>
<tr>
<td>pilot</td>
<td>1</td>
<td>31</td>
<td>5.81e-09</td>
<td>-5.5748969768e+02</td>
<td>33.20</td>
</tr>
<tr>
<td>pilot4</td>
<td>1</td>
<td>30</td>
<td>2.28e-09</td>
<td>-2.5811392580e+03</td>
<td>3.81</td>
</tr>
</tbody>
</table>

Table A.2: Set of 95 Netlib problems solved under the default tolerance setting of $10^{-8}$
<table>
<thead>
<tr>
<th>Problem</th>
<th>Converged</th>
<th>Iter</th>
<th>TRE</th>
<th>Obj. Value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>pilot87</td>
<td>1</td>
<td>38</td>
<td>8.82e-09</td>
<td>3.0171040806e+02</td>
<td>128.15</td>
</tr>
<tr>
<td>pilotja</td>
<td>1</td>
<td>31</td>
<td>8.06e-10</td>
<td>-6.1131364644e+03</td>
<td>7.94</td>
</tr>
<tr>
<td>pilotnov</td>
<td>1</td>
<td>20</td>
<td>2.61e-13</td>
<td>-4.4972761882e+03</td>
<td>4.74</td>
</tr>
<tr>
<td>pilotwe</td>
<td>1</td>
<td>37</td>
<td>2.49e-13</td>
<td>-2.7201075328e+06</td>
<td>5.44</td>
</tr>
<tr>
<td>recipe</td>
<td>1</td>
<td>9</td>
<td>6.66e-09</td>
<td>-2.666159983e+02</td>
<td>0.38</td>
</tr>
<tr>
<td>sc105</td>
<td>1</td>
<td>10</td>
<td>1.46e-11</td>
<td>-5.2202061212e+01</td>
<td>0.25</td>
</tr>
<tr>
<td>sc205</td>
<td>1</td>
<td>10</td>
<td>9.75e-11</td>
<td>-5.2202061198e+01</td>
<td>0.31</td>
</tr>
<tr>
<td>sc50a</td>
<td>1</td>
<td>10</td>
<td>4.78e-15</td>
<td>-6.4575077059e+01</td>
<td>0.22</td>
</tr>
<tr>
<td>sc50b</td>
<td>1</td>
<td>7</td>
<td>5.80e-09</td>
<td>-6.9999999800e+01</td>
<td>0.19</td>
</tr>
<tr>
<td>scagr25</td>
<td>1</td>
<td>17</td>
<td>8.64e-11</td>
<td>-1.4753433061e+07</td>
<td>0.75</td>
</tr>
<tr>
<td>scagr7</td>
<td>1</td>
<td>14</td>
<td>2.08e-11</td>
<td>-2.3313898243e+06</td>
<td>0.36</td>
</tr>
<tr>
<td>scfxm1</td>
<td>1</td>
<td>19</td>
<td>1.91e-14</td>
<td>1.8416759028e+04</td>
<td>0.99</td>
</tr>
<tr>
<td>scfxm2</td>
<td>1</td>
<td>21</td>
<td>4.59e-14</td>
<td>3.6660261565e+04</td>
<td>1.86</td>
</tr>
<tr>
<td>scfxm3</td>
<td>1</td>
<td>21</td>
<td>1.69e-13</td>
<td>5.4901254550e+04</td>
<td>2.62</td>
</tr>
<tr>
<td>scorpion</td>
<td>1</td>
<td>15</td>
<td>1.69e-13</td>
<td>1.8781248227e+03</td>
<td>0.60</td>
</tr>
<tr>
<td>scrs8</td>
<td>1</td>
<td>24</td>
<td>7.23e-12</td>
<td>9.0429695381e+02</td>
<td>1.37</td>
</tr>
<tr>
<td>scsd1</td>
<td>1</td>
<td>9</td>
<td>8.07e-09</td>
<td>8.6666720415e+00</td>
<td>0.39</td>
</tr>
<tr>
<td>scsd6</td>
<td>1</td>
<td>11</td>
<td>1.24e-09</td>
<td>5.050001491e+01</td>
<td>0.67</td>
</tr>
<tr>
<td>scsd8</td>
<td>1</td>
<td>11</td>
<td>1.44e-11</td>
<td>9.0499999995e+02</td>
<td>1.18</td>
</tr>
<tr>
<td>sctap1</td>
<td>1</td>
<td>17</td>
<td>2.48e-13</td>
<td>1.4122500000e+03</td>
<td>0.68</td>
</tr>
<tr>
<td>sctap2</td>
<td>1</td>
<td>19</td>
<td>5.14e-13</td>
<td>1.7248071429e+03</td>
<td>2.12</td>
</tr>
<tr>
<td>sctap3</td>
<td>1</td>
<td>18</td>
<td>2.56e-14</td>
<td>1.4240000000e+03</td>
<td>2.60</td>
</tr>
<tr>
<td>seba</td>
<td>1</td>
<td>22</td>
<td>6.67e-10</td>
<td>1.5711600000e+04</td>
<td>4.58</td>
</tr>
<tr>
<td>share1b</td>
<td>1</td>
<td>22</td>
<td>8.65e-15</td>
<td>-7.6589318579e+04</td>
<td>0.65</td>
</tr>
<tr>
<td>share2b</td>
<td>1</td>
<td>13</td>
<td>5.32e-13</td>
<td>-4.1573224074e+02</td>
<td>0.37</td>
</tr>
</tbody>
</table>

Table A.3: Set of 95 Netlib problems solved under the default tolerance setting of $10^{-8}$
<table>
<thead>
<tr>
<th>Problem</th>
<th>Converged</th>
<th>Iter</th>
<th>TRE</th>
<th>Obj. Value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>shell</td>
<td>1</td>
<td>21</td>
<td>8.92e-15</td>
<td>1.2088253460e+09</td>
<td>1.53</td>
</tr>
<tr>
<td>ship04l</td>
<td>1</td>
<td>14</td>
<td>3.25e-14</td>
<td>1.7933245380e+06</td>
<td>1.27</td>
</tr>
<tr>
<td>ship04s</td>
<td>1</td>
<td>14</td>
<td>1.20e-10</td>
<td>1.7987147005e+06</td>
<td>0.91</td>
</tr>
<tr>
<td>ship08l</td>
<td>1</td>
<td>16</td>
<td>3.16e-12</td>
<td>1.9090552114e+06</td>
<td>2.62</td>
</tr>
<tr>
<td>ship08s</td>
<td>1</td>
<td>15</td>
<td>3.77e-13</td>
<td>1.9200982105e+06</td>
<td>1.34</td>
</tr>
<tr>
<td>ship12l</td>
<td>1</td>
<td>18</td>
<td>5.15e-11</td>
<td>1.4701879193e+06</td>
<td>3.48</td>
</tr>
<tr>
<td>ship12s</td>
<td>1</td>
<td>18</td>
<td>1.48e-11</td>
<td>1.4892361344e+06</td>
<td>1.65</td>
</tr>
<tr>
<td>sierra</td>
<td>1</td>
<td>17</td>
<td>8.59e-11</td>
<td>1.5394362184e+07</td>
<td>3.60</td>
</tr>
<tr>
<td>stair</td>
<td>1</td>
<td>14</td>
<td>2.44e-10</td>
<td>-2.5126695116e+02</td>
<td>1.27</td>
</tr>
<tr>
<td>standata</td>
<td>1</td>
<td>17</td>
<td>1.45e-14</td>
<td>1.2576995000e+03</td>
<td>1.16</td>
</tr>
<tr>
<td>standgub</td>
<td>1</td>
<td>17</td>
<td>4.50e-13</td>
<td>1.2576995000e+03</td>
<td>1.18</td>
</tr>
<tr>
<td>standmps</td>
<td>1</td>
<td>24</td>
<td>1.35e-13</td>
<td>1.4060175000e+03</td>
<td>1.80</td>
</tr>
<tr>
<td>stocfor1</td>
<td>1</td>
<td>16</td>
<td>2.46e-11</td>
<td>-4.1131983232e+04</td>
<td>0.41</td>
</tr>
<tr>
<td>stocfor2</td>
<td>1</td>
<td>22</td>
<td>6.82e-09</td>
<td>-3.9024399857e+04</td>
<td>3.88</td>
</tr>
<tr>
<td>stocfor3</td>
<td>1</td>
<td>32</td>
<td>2.01e-10</td>
<td>-3.9976783941e+04</td>
<td>40.59</td>
</tr>
<tr>
<td>truss</td>
<td>1</td>
<td>19</td>
<td>3.72e-11</td>
<td>4.5881584719e+05</td>
<td>6.21</td>
</tr>
<tr>
<td>tuff</td>
<td>1</td>
<td>20</td>
<td>6.22e-09</td>
<td>2.9215032328e-01</td>
<td>1.64</td>
</tr>
<tr>
<td>vtpbase</td>
<td>1</td>
<td>23</td>
<td>2.81e-11</td>
<td>1.2983146246e+05</td>
<td>0.95</td>
</tr>
<tr>
<td>wood1p</td>
<td>1</td>
<td>19</td>
<td>3.73e-09</td>
<td>1.4429031381e+00</td>
<td>11.25</td>
</tr>
<tr>
<td>woodw</td>
<td>1</td>
<td>28</td>
<td>6.85e-10</td>
<td>1.3044763953e+00</td>
<td>11.28</td>
</tr>
</tbody>
</table>

Table A.4: Set of 95 Netlib problems solved under the default tolerance setting of $10^{-8}$
Bibliography


