A Stable Primal-Dual Approach for Linear Programming under Nondegeneracy Assumptions

Maria Gonzalez-Lima * Hua Wei† Henry Wolkowicz ‡

December 13, 2007

University of Waterloo
Department of Combinatorics & Optimization
Waterloo, Ontario N2L 3G1, Canada
Research Report CORR 2004-26
COAP 1172-04

Key words: Linear Programming, large sparse problems, preconditioned conjugate gradients, stability.

Abstract

This paper studies a primal-dual interior/exterior-point path-following approach for linear programming that is motivated on using an iterative solver rather than a direct solver for the search direction. We begin with the usual perturbed primal-dual optimality equations. Under nondegeneracy assumptions, this nonlinear system is well-posed, i.e. it has a nonsingular Jacobian at optimality and is not necessarily ill-conditioned as the iterates approach optimality. Assuming that a basis matrix (easily factorizable and well-conditioned) can be found, we apply a simple preprocessing step to eliminate both the primal and dual feasibility equations. This results in a single bilinear equation that maintains the well-posedness property. Sparsity is mantained. We then apply either a direct solution method or an iterative solver (within an inexact Newton framework) to solve this equation. Since the linearization is well posed, we use affine scaling and do not maintain nonnegativity once we are close enough

*Research supported by Universidad Simón Bolívar (DID-GID001) and Conicit (project G97000592), Venezuela. E-mail mgl@cesma.usb.ve

†Research supported by The Natural Sciences and Engineering Research Council of Canada and Bell Canada. E-mail h3wei@math.uwaterloo.ca

‡Research supported by The Natural Sciences and Engineering Research Council of Canada. E-mail hwolkowicz@uwaterloo.ca

[0]URL for paper: orion.math.uwaterloo.ca/~hwolkowi/henry/reports/ABSTRACTS.html This report is a revision of the earlier CORR 2001-66.

to the optimum, i.e. we apply a *change to a pure Newton step* technique. In addition, we correctly identify some of the primal and dual variables that converge to 0 and delete them (purify step).

We test our method with random nondegenerate problems and problems from the Netlib set, and we compare it with the standard Normal Equations ***NEQ*** approach. We use a heuristic to find the basis matrix. We show that our method is efficient for large, well-conditioned problems. It is slower than ***NEQ*** on ill-conditioned problems, but it yields higher accuracy solutions.

Contents

List of Tables

List of Figures

1 Introduction

The purpose of this paper is to study an alternative primal-dual path-following approach for Linear Programming (***LP***) that is based on an (inexact) Newton method with preconditioned conjugate gradients (PCG). We do not form the usual *normal equations* system, i.e. no ill-conditioned system is formed. For well-conditioned problems with special structure, our approach exploits sparsity and obtains high accuracy solutions.

The primal ***LP*** we consider is

$$(\boldsymbol{LP}) \qquad \begin{array}{rll} p^* := & \min & c^T x \\ & \text{s.t.} & Ax = b \\ & & x \geq 0. \end{array} \tag{1.1}$$

The dual program is

$$(\boldsymbol{DLP}) \qquad \begin{array}{rll} d^* := & \max & b^T y \\ & \text{s.t.} & A^T y + z = c \\ & & z \geq 0. \end{array} \tag{1.2}$$

Here $A \in \Re^{m \times n}$, $c \in \Re^n$, $b \in \Re^m$. We assume that $m < n$, A has full rank, and the set of strictly feasible points defined as

$$\mathcal{F}^+ = \{(x, y, z) : Ax = b, A^T y + z = c, x > 0, z > 0\}$$

is not empty. Our algorithm assumes that we can obtain the special structure $A = (\, B \quad E \,)$ (perhaps by permuting rows and columns), where B is $m \times m$, nonsingular, not ill-conditioned, and it is inexpensive to solve a linear system with B. Our approach is most efficient under nondegeneracy assumptions.

Throughout this paper we use the following notation. Given a vector $x \in \Re^n$, the matrix $X \in \Re^{n \times n}$, or equivalently $\mathrm{Diag}\,(x)$, denotes the diagonal matrix with the vector x on the diagonal. The vector e denotes the vector of all ones (of appropriate dimension) and I denotes the identity matrix, also with the appropriate correct dimension. Unless stated otherwise, $\|.\|$ denotes the Euclidean norm. And, given $F : \Re^n \to \Re^n$, we let $F'(x)$ denote the Jacobian of F at x.

1.1 Background and Motivation

Solution methods for Linear Programming (***LP***) have evolved dramatically following the introduction of interior point methods. (For a historical development, see e.g. [45, 50] and the references therein.) Currently the most popular methods are the elegant primal-dual path-following

methods. These methods are based on log-barrier functions applied to the nonnegativity constraints. For example, we can start with the dual log-barrier problem, with parameter $\mu > 0$,

$$\text{(Dlogbarrier)} \qquad \begin{array}{rll} d_\mu^* := & \max & b^T y + \mu \sum_{j=1}^n \log z_j \\ & \text{s.t.} & A^T y + z = c \\ & & z > 0. \end{array} \tag{1.3}$$

The stationary point of the Lagrangian for (1.3) (x plays the role of the vector of Lagrange multipliers for the equality constraints) yields the optimality conditions

$$\begin{pmatrix} A^T y + z - c \\ Ax - b \\ X - \mu Z^{-1} \end{pmatrix} = 0, \quad x, z > 0. \tag{1.4}$$

For each $\mu > 0$, the solution of these optimality conditions is unique. The set of these solutions forms the so-called *central path* that leads to the optimum of ($\boldsymbol{LP}$) as μ tends to 0. However, it is well known that the Jacobian of these optimality conditions grows ill-conditioned as the log-barrier parameter μ approaches 0. This ill-conditioning (as observed for general nonlinear programs in the classical [19]) can be avoided by changing the third row of the optimality conditions to the more familiar form of the complementary slackness conditions, $ZXe - \mu e = 0$. One then applies a damped Newton method to solve this system while maintaining positivity of x, z and reducing μ to 0. Equivalently, this can be viewed as an *interior-point* method with *path-following* of the central path.

It is inefficient to solve the resulting linearized system as it stands, since it has special structure that can be exploited. Block eliminations yield a positive definite system (called the normal equations, $\boldsymbol{NEQ}$) of size m, with matrix ADA^T, where D is diagonal; see Section 2.2. Alternatively, a larger *augmented system* or *quasi-definite system*, of size $(m+n) \times (m+n)$ can be used, e.g. [51], [45, Chap. 19]. However, the ill-conditioning returns in both cases, i.e. we first get rid of the ill-conditioning by changing the log-barrier optimality conditions; we then bring it back with the backsolves after the block eliminations; see Section 2.2.2. Another potential difficulty is the possible loss of sparsity in forming ADA^T.

However, there are advantages when considering the two reduced systems. The size of the normal equations system is m compared to the size $m+2n$ of the original linearized system. And efficient factorization schemes can be applied. The augmented system is larger but there are gains in exploiting sparsity when applying factorization schemes. Moreover, the ill-conditioning for both systems has been carefully studied. For example, the idea of *structured singularity* is used in [49] to show that the normal equations for nonlinear programming can be solved in a stable way in a neighbourhood of the central path. However, the backsolve step can still be negatively affected by ill-conditioning if the assumptions in [49] are not satisfied; see our Example 2.2 below. In particular, the assumption of positive definiteness of the Hessian in [49] does not apply to $\boldsymbol{LP}$. For further results on the ill-conditioning of the normal equations and

the augmented system, see e.g. [49, 52, 53] and the books [45, 50]. For a discussion on the growth in the condition number after the backsolve, see Remark 2.6 below.

The major work (per iteration) is the formation and factorization of the reduced system. However, factorization schemes can fail for huge problems and/or problems where the reduced system is not sparse. If A is sparse, then one could apply conjugate-gradient type methods and avoid the matrix multiplications, e.g. one could use $A(D(A^T v))$ for the matrix-vector multiplications for the ADA^T system. However, classical iterative techniques for large sparse linear systems have not been generally used. One difficulty is that the normal equations can become ill-conditioned. Iterative schemes need efficient preconditioners to be competitive. This can be the case for problems with special structure, see e.g. [27, 35]. For other iterative approaches see e.g. [14, 31, 2, 26, 34, 37, 8, 13].

Although the reduced normal equations approach has benefits as mentioned above, the ill conditioning that arises for ***NEQ*** and during the backsolve step is still a potential numerical problem for obtaining high accuracy solutions. In this paper we look at a modified approach for these interior point methods. We use a simple preprocessing technique to eliminate the primal and dual feasibility equations. Under nondegeneracy assumptions, the result is a bilinear equation that does not necessarily result in a linearized ill-conditioned system. (Though the size of our linearized system is $n \times n$ compared to $m \times m$ for ***NEQ***.) Moreover, in contrast to ***NEQ***, the backsolve steps are stable. Therefore we can use this stable linear system to find the Newton search direction within a primal-dual interior point framework. Furthermore, this allows for modifications in the primal-dual interior point framework, e.g. we do not have to always backtrack from the boundary and stay strictly interior. We then work on this linear system with an inexact Newton approach and use a preconditioned conjugate-gradient-type method to (approximately) solve the linearized system for the search direction. One can still use efficient Cholesky techniques in the preconditioning process, e.g. partial Cholesky factorizations that preserve sparsity (or partial QR factorizations). The advantage is that these techniques are applied to a system that does not necessarily get ill-conditioned and sparsity can be directly exploited without using special techniques. As in the case mentioned above, the approach is particularly efficient when the structure of the problem can be exploited to construct efficient preconditioners. (This is the case for certain classes of Semidefinite Programming (SDP) problems, see [48].) We also use a change to a pure Newton step and purification techniques to speed up the convergence. In particular, the robustness of the linear system allows us to apply the so-called Tapia indicators [18] to correctly detect those variables that are zero at the solution.

1.2 Outline and Main Contributions

In Section 2 we introduce the basic properties for ***LP*** interior point methods. Section 2.2 presents the block elimination scheme for ***NEQ*** system, i.e. the scheme to find the normal equations, ***NEQ***. This is compared to the block elimination scheme for our *stable method* in Section 2.4. In particular, we show that, as we approach the optimum, the condition number for

the ***NEQ*** system converges to infinity while (under nondegeneracy assumptions) the condition number for the stable method stays uniformly bounded. This is without any special assumptions on the step lengths during the iterations, see Proposition 2.5. In fact, the reciprocal of the condition number for the ***NEQ*** system is $O(\mu)$, see Remark 2.6. (In [25] it is shown that the condition number of the normal equations matrix (not the entire system) stays uniformly bounded under the nondegeneracy assumption and neighbourhood type restrictions on the step lengths.) We include numerical examples that illustrate numerical roundoff difficulties. In Section 3 we present the primal-dual interior point algorithm. The preconditioning techniques are given in Section 3.2. The change to a pure Newton step technique is described in Section 3.3 while the purification technique appears in Section 3.4. The numerical tests, on randomly generated problems and the standard NETLIB test set, are given in Section 4; concluding remarks are given in Section 5.

2 Duality, Optimality, and Block Eliminations

We first summarize the well known duality properties for ***LP*** . If both primal and dual problems have feasible solutions, x, y, z, then: $c^T x \geq b^T y$ (weak duality); and $p^* = d^*$ and both optimal values are attained (strong duality).

The well known primal-dual optimality conditions (primal feasibility, dual feasibility, and complementary slackness) follow from the weak and strong duality properties.

Theorem 2.1 *The primal-dual variables (x, y, z), with $x, z \geq 0$, are optimal for the primal-dual pair of **LP** s if and only if*

$$F(x, y, z) := \begin{pmatrix} A^T y + z - c \\ Ax - b \\ ZXe \end{pmatrix} = 0. \tag{2.1}$$

■

Moreover, for *feasible* (x, y, z), we get

$$\begin{aligned} \text{duality gap} &= c^T x - b^T y \\ &= x^T \left(c - A^T y\right) \\ &= x^T z. \end{aligned} \tag{2.2}$$

2.1 Linearization

Note that $F : \Re^n \times \Re^m \times \Re^n \to \Re^n \times \Re^m \times \Re^n$. Let $\mu > 0$ and let us consider the perturbed optimality conditions

$$F_\mu(x, y, z) := \begin{pmatrix} A^T y + z - c \\ Ax - b \\ ZXe - \mu e \end{pmatrix} = \begin{pmatrix} r_d \\ r_p \\ r_c \end{pmatrix} = 0, \tag{2.3}$$

thus defining the dual and primal residual vectors r_d, r_p and perturbed complementary slackness r_c. Currently, the successful primal-dual algorithms are path-following algorithms that use a damped Newton method to solve this system approximately with $(x, z) > 0$. This is done in conjunction with decreasing μ to 0. The Newton equation (the linearization) for the Newton direction $\Delta s = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}$ is

$$F'_\mu(x, y, z)\Delta s = \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & 0 & X \end{pmatrix} \Delta s = -F_\mu(x, y, z). \tag{2.4}$$

Damped Newton steps

$$x \leftarrow x + \alpha_p \Delta x, \quad y \leftarrow y + \alpha_d \Delta y, \quad z \leftarrow z + \alpha_d \Delta z,$$

are taken that *backtrack* from the nonnegativity boundary to maintain the positivity/interiority, $x > 0, z > 0$.

Suppose that $F_\mu(x, y, z) = 0$ in (2.3). Then (2.3) and (2.2) imply

$$\mu = \frac{1}{n}\mu e^T e = \frac{1}{n} e^T ZXe = \frac{1}{n} z^T x = \frac{1}{n}(\text{ duality gap}),$$

i.e. the barrier parameter μ is a good measure of the duality gap. However, most practical interior-point methods are infeasible methods, i.e. they do not start with primal-dual feasible solutions and stop with nonzero residuals. Similarly, if feasibility is obtained, roundoff error can result in nonzero residuals r_d, r_p in the next iteration. Therefore, in both cases,

$$\begin{aligned} n\mu &= z^T x \\ &= (c - A^T y + r_d)^T x \\ &= \left(c^T x - y^T Ax + r_d^T x\right) \\ &= \left(c^T x - y^T (b + r_p) + r_d^T x\right) \\ &= \left(c^T x - b^T y - r_p^T y + r_d^T x\right) \\ &= (c + r_d)^T x - (b + r_p)^T y, \end{aligned} \tag{2.5}$$

i.e. $n\mu$ measures the duality gap of a perturbed $\boldsymbol{LP}$. (See e.g. the survey article on error bounds [40].)

2.2 Reduction to the Normal Equations

The Newton equation (2.4) is solved at each iteration of a primal-dual interior point (p-d i-p) algorithm. This is the major work involved in these path-following algorithms. Solving (2.4) directly is too expensive. There are several manipulations that can be done that result in a much smaller system. We can consider this in terms of block elimination steps.

2.2.1 First Step in Block Elimination for Normal Equations

The customary first step in the literature is to eliminate Δz using the first row of equations. (Note the linearity and coefficient I for z in the first row of (2.3).) Equivalently, apply elementary row operations to matrix $F'_\mu(x,y,z)$, or find a matrix P_Z such that the multiplication of $P_Z F'_\mu(x,y,z)$ results in a matrix with the corresponding columns of Δz being formed by the identity matrix and zero matrices. This is,

$$\begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -X & 0 & I \end{pmatrix} \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & 0 & X \end{pmatrix} = \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & -XA^T & 0 \end{pmatrix}, \tag{2.6}$$

with right-hand side

$$-\begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -X & 0 & I \end{pmatrix} \begin{pmatrix} A^T y + z - c \\ Ax - b \\ ZXe - \mu e \end{pmatrix} = -\begin{pmatrix} r_d \\ r_p \\ -Xr_d + ZXe - \mu e \end{pmatrix}. \tag{2.7}$$

We let

$$P_Z = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -X & 0 & I \end{pmatrix}, \quad K = \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & -XA^T & 0 \end{pmatrix}. \tag{2.8}$$

2.2.2 Second Step in Block Elimination for Normal Equations

The so-called normal equations are obtained by further eliminating Δx. (Note the **non**linearity in x in the third row of (2.3).) Following a similar procedure, we define the matrices F_n, P_n with

$$\begin{aligned} F_n := P_n K &:= \begin{pmatrix} I & 0 & 0 \\ 0 & I & -AZ^{-1} \\ 0 & 0 & Z^{-1} \end{pmatrix} \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & -XA^T & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & A^T & I_n \\ 0 & \boxed{AZ^{-1}XA^T} & 0 \\ I_n & -Z^{-1}XA^T & 0 \end{pmatrix}. \end{aligned} \tag{2.9}$$

The right-hand side becomes

$$-P_nP_Z\begin{pmatrix} A^Ty+z-c \\ Ax-b \\ ZXe-\mu e \end{pmatrix} = \begin{pmatrix} -r_d \\ -r_p+A(-Z^{-1}Xr_d+x-\mu Z^{-1}e) \\ Z^{-1}Xr_d-x+\mu Z^{-1}e \end{pmatrix}. \tag{2.10}$$

The algorithm for finding the Newton search direction using the normal equations is now evident from (2.9): we move the third column before column one and interchange the second and third rows:

$$\begin{pmatrix} I_n & 0 & A^T \\ 0 & I_n & -Z^{-1}XA^T \\ 0 & 0 & \boxed{AZ^{-1}XA^T} \end{pmatrix}\begin{pmatrix} \Delta z \\ \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} -r_d \\ Z^{-1}Xr_d-x+\mu Z^{-1}e \\ -r_p+A(-Z^{-1}Xr_d+x-\mu Z^{-1}e) \end{pmatrix}. \tag{2.11}$$

Thus we first solve for Δy, then backsolve for Δx, and finally backsolve for Δz. This block upper-triangular system has the disadvantage of being ill-conditioned when evaluated at points close to the optimum. This will be shown in the next section. The condition number for the system is found from the condition number of the matrix F_n and not just the matrix $AZ^{-1}XA^T$. (Though, as mentioned above, the latter can have a uniformly bounded condition number under some standard neighbourhood assumptions plus the nondegeneracy assumption, see e.g. [25].) F_n is unnecessarily ill-conditioned because P_n is an ill-conditioned transformation.

2.3 Roundoff Difficulties for NEQ Examples

We present several numerical examples with ***NEQ*** (cases that are not covered in [49]) involving combinations of: degeneracy or nondegeneracy; feasible or infeasible starting points; and large residuals. (Difficulties with degeneracy and ***NEQ*** appear in e.g. Figure 4.1 below.)

2.3.1 Nondegenerate but with Large Residual

Even if a problem is nondegenerate, difficulties can arise if the current primal-dual point has a large residual error relative to the duality gap. This emphasizes the importance of keeping the iterates *well-centered* for ***NEQ***.

Example 2.2 *Here the residuals are not the same order as μ. We see that we get catastrophic roundoff error. Consider the simple data*

$$A=\begin{pmatrix}1 & 1\end{pmatrix},\ c=\begin{pmatrix}-1\\1\end{pmatrix},\ b=1.$$

The optimal primal-dual variables are

$$x=\begin{pmatrix}1\\0\end{pmatrix},\ y=-1,\ z=\begin{pmatrix}0\\2\end{pmatrix}.$$

We use 6 *decimals accuracy in the arithmetic and start with the following points (nonfeasible) obtained after several iterations:*

$$x = \begin{pmatrix} 9.183012 \times 10^{-1} \\ 1.356397 \times 10^{-8} \end{pmatrix}, \; z = \begin{pmatrix} 2.193642 \times 10^{-8} \\ 1.836603 \end{pmatrix}, \; y = -1.163398 \; .$$

The residuals (relatively large) and duality gap measure are:

$$\|r_b\| = 0.081699, \quad \|r_d\| = 0.36537, \quad \mu = x^T z/n = 2.2528 \times 10^{-8}.$$

Though μ *is small, we still have large residuals for both primal and dual feasibility. Therefore,* $2\mu = n\mu$ *is not a true measure of the duality gap. The two search directions,* $\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}$*, that are found using first the full matrix* F'_μ *in* (2.4)*, and second the system with* F_n *in* (2.9) *(solving* Δy *first and then backsolving* Δx*,* Δz*) are, respectively,*

$$\begin{pmatrix} 8.16989 \times 10^{-2} \\ -1.35442 \times 10^{-8} \\ 1.63400 \times 10^{-1} \\ -2.14348 \times 10^{-8} \\ 1.63400 \times 10^{-1} \end{pmatrix}, \quad \begin{pmatrix} -6.06210 \times 10 \\ -1.35441 \times 10^{-8} \\ 1.63400 \times 10^{-1} \\ 0 \\ 1.63400 \times 10^{-1} \end{pmatrix}.$$

Though the error in Δy *is small, since* $m = 1$*, the error after the backsubstitution for the first component of* Δx *is large, with no decimals accuracy. The resulting search direction results in no improvements in the residuals or the duality gap. Using the accurate direction from* F_s*, see* (2.16) *below, results in good improvement and convergence.*

In practice, the residuals generally decrease at the same rate as μ*. (For example, this is assumed in the discussion in [51].) But, as our tests in Section 4 below show, the residuals and roundoff do cause a problem for* ***NEQ*** *when* μ *gets small.*

2.3.2 Degenerate Case

We use the data

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \quad c = (\, 1 \quad 1 \quad 1 \quad 1 \,)^T . \tag{2.12}$$

An optimal primal-dual solution is

$$x^* = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad y^* = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad z^* = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

This problem is degenerate; $x = [2,\ 0,\ 0,\ 0]^T$ is also an optimal solution. We partition into index sets $\mathcal{B} = \{1, 3\}$ and $\mathcal{N} = \{2, 4\}$. Following the analysis in [53], we assume that $x,\ z$ are in a certain neighbourhood of the central path and that the residuals are of order μ. Then the computed values satisfy (again from [53])

$$\begin{aligned} \widehat{\Delta y} - \Delta y &= O(\mathbf{u}); \\ \widehat{\Delta x_{\mathcal{B}}} - \Delta x_{\mathcal{B}} = O(\mathbf{u});\ \ \widehat{\Delta x_{\mathcal{N}}} - \Delta x_{\mathcal{N}} &= O(\mu\mathbf{u}); \\ \widehat{\Delta z_{\mathcal{B}}} - \Delta z_{\mathcal{B}} = O(\mu\mathbf{u});\ \ \widehat{\Delta z_{\mathcal{N}}} - \Delta z_{\mathcal{N}} &= O(\mathbf{u}). \end{aligned} \tag{2.13}$$

Here $\hat{\cdot}$ denotes the computed solution, and $\mathbf{u}$ is the unit roundoff. The results (2.13) hold independent of the condition number of the system. Furthermore, the analysis in [53] implies that the computed solutions progress well, i.e. with step lengths close to one.

We now present two degenerate examples where the bounds (2.13) fail for ***NEQ*** .

We first present a pair of x and z that satisfy our assumptions (i.e. they are close to the central path and the infeasibility residuals are $O(\mu)$). We use MATLAB's "\" (double precision) solver on the full system

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} -r_d \\ -r_p \\ -ZXe + \mu e \end{pmatrix} \tag{2.14}$$

and consider this to be the *accurate* evaluation of the search direction. We then compare this with the ***NEQ*** approach, i.e. we solve

$$\begin{pmatrix} 0 & A^T & I_n \\ 0 & AZ^{-1}XA^T & 0 \\ I_n & -Z^{-1}XA^T & 0 \end{pmatrix} \begin{pmatrix} \widehat{\Delta x} \\ \widehat{\Delta y} \\ \widehat{\Delta z} \end{pmatrix} = \begin{pmatrix} -r_d \\ -r_p + A(-Z^{-1}Xr_d + x - \mu Z^{-1}e) \\ Z^{-1}Xr_d - x + \mu Z^{-1}e \end{pmatrix}.$$

We solve $\widehat{\Delta y}$ first, and then backsolve for $\widehat{\Delta x}$ and $\widehat{\Delta z}$. We simulate the $fl(\cdot)$ operation by keeping the 8 most significant digits after each arithmetic operation.

Example 2.3 *We start with infeasible x and z*

$$x = \begin{pmatrix} 9.9985999 \times 10^{-1} \\ 2.3975770 \times 10^{-4} \\ 9.9983748 \times 10^{-1} \\ 1.7333628 \times 10^{-4} \end{pmatrix}, \quad z = \begin{pmatrix} 1.3758855 \times 10^{-4} \\ 9.9979802 \times 10^{-1} \\ 2.8397156 \times 10^{-4} \\ 1.0001754 \end{pmatrix}$$

obtained by perturbing the optimal x^* and z^*. We get

$$\mu = 2.1 \times 10^{-4}, \quad r_p = \begin{pmatrix} -3.0 \times 10^{-4} \\ 3.1 \times 10^{-4} \end{pmatrix}, \quad r_d = \begin{pmatrix} -4.2 \times 10^{-5} \\ 1.8 \times 10^{-4} \\ 1.0 \times 10^{-4} \\ -1.7 \times 10^{-5} \end{pmatrix}.$$

Therefore the residuals are of order μ. The solutions for Δy satisfy

$$\Delta y = \begin{pmatrix} -2.9255369 \times 10^{-5} \\ -1.8441334 \times 10^{-1} \end{pmatrix}, \quad \widehat{\Delta} y = \begin{pmatrix} -2.9262363 \times 10^{-5} \\ -1.8441335 \times 10^{-1} \end{pmatrix},$$
$$\Delta y - \widehat{\Delta} y = \begin{pmatrix} 6.99389248 \times 10^{-9} \\ 5.29186195 \times 10^{-9} \end{pmatrix}.$$

Since the system for Δy is diagonal, the error is approximately equal to the unit roundoff, 10^{-8}. But the backsolve step

$$\Delta x = Z^{-1} X r_d - x + \mu Z^{-1} e + Z^{-1} X A^T \Delta y$$

is inaccurate because P_n in (2.9) was an ill-conditioned transformation:

$$\Delta x = \begin{pmatrix} 1.92649415 \times 10^{-4} \\ -1.19476143 \times 10^{-4} \\ 1.098805846 \times 10^{-4} \\ 6.722683477 \times 10^{-5} \end{pmatrix}, \quad \widehat{\Delta x} = \begin{pmatrix} 1.5234654 \times 10^{-4} \\ -1.1947615 \times 10^{-4} \\ 1.5017835 \times 10^{-4} \\ 6.7226831 \times 10^{-5} \end{pmatrix},$$
$$\Delta x - \widehat{\Delta x} = \begin{pmatrix} 4.0302875 \times 10^{-5} \\ 7.3832399 \times 10^{-12} \\ -4.0297765 \times 10^{-5} \\ 3.7664799 \times 10^{-12} \end{pmatrix}.$$

Although the nonbasic variables have absolute error $O(\mu \mathbf{u})$, this is not true for the basic variables, where we get approximately $O(\frac{\mathbf{u}}{\mu})$. (In terms of relative error, it is $O(\frac{\mathbf{u}}{\mu^2})$, since $(\Delta x, \Delta y, \Delta z)$ is $O(\mu)$.)

Example 2.4 *This second example shows that catastrophic error can occur in $\widehat{\Delta y}$. In this example, we change the data matrix A to*

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 2 & 2 & 2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \quad c = (\, 1 \quad 1 \quad 1 \quad 1 \,)^T. \tag{2.15}$$

An optimal solution is

$$x^* = (\, 1 \quad 0 \quad 1 \quad 0 \,)^T, \quad z^* = (\, 0 \quad 1 \quad 0 \quad 1 \,)^T.$$

We let the initial x and z be

$$x = \begin{pmatrix} 9.9985681 \times 10^{-1} \\ 8.1713298 \times 10^{-5} \\ 1.0001432 \\ 1.634266 \times 10^{-4} \end{pmatrix}, \quad z = \begin{pmatrix} 1.9454628 \times 10^{-4} \\ 9.9961681 \times 10^{-1} \\ 1.9454628 \times 10^{-4} \\ 1.0001916 \end{pmatrix}.$$

Again, we check the duality gap parameter and the residuals:

$$\mu = 2.1 \times 10^{-4}, \quad r_p = \begin{pmatrix} 9.99999994 \times 10^{-9} \\ 1.99959995 \times 10^{-8} \end{pmatrix},$$
$$r_d = \begin{pmatrix} 4.77999995 \times 10^{-9} \\ -1.50000001 \times 10^{-9} \\ 4.77999995 \times 10^{-9} \\ 5.75000003 \times 10^{-9} \end{pmatrix}.$$

In this case $\widehat{\Delta y}$ *is quite inaccurate:*

$$\Delta y = \begin{pmatrix} 6.47338334 \times 10^{-1} \\ -3.23651175 \times 10^{-1} \end{pmatrix}, \quad \widehat{\Delta y} = \begin{pmatrix} -1.5866402 \times 10^{-1} \\ 7.935 \times 10^{-2} \end{pmatrix},$$
$$\Delta y - \widehat{\Delta y} = \begin{pmatrix} 8.060023536 \times 10^{-1} \\ -4.030011751 \times 10^{-1} \end{pmatrix}.$$

For Δx *we have*

$$\Delta x = \begin{pmatrix} 1.16701057 \times 10^{-4} \\ 2.39921125 \times 10^{-5} \\ -1.16711057 \times 10^{-4} \\ 4.79842209 \times 10^{-5} \end{pmatrix}, \quad \widehat{\Delta x} = \begin{pmatrix} 7.4739018 \times 10^{-5} \\ 8.9878474 \times 10^{-5} \\ -1.5868482 \times 10^{-4} \\ -1.7864276 \times 10^{-5} \end{pmatrix},$$
$$\Delta x - \widehat{\Delta x} = \begin{pmatrix} 4.196203945 \times 10^{-5} \\ -6.588636152 \times 10^{-5} \\ 4.197376255 \times 10^{-5} \\ 6.584849696 \times 10^{-5} \end{pmatrix}.$$

For Δz *we have*

$$\Delta z = \begin{pmatrix} -3.59881646 \times 10^{-5} \\ 6.4730235 \times 10^{-1} \\ -3.5988165 \times 10^{-5} \\ -3.2365118 \times 10^{-1} \end{pmatrix}, \quad \widehat{\Delta z} = \begin{pmatrix} -3.598 \times 10^{-5} \\ -1.587 \times 10^{-1} \\ -3.598 \times 10^{-5} \\ 7.935 \times 10^{-2} \end{pmatrix},$$
$$\Delta z - \widehat{\Delta z} = \begin{pmatrix} -8.16462922 \times 10^{-9} \\ 8.06002352 \times 10^{-1} \\ -8.16462922 \times 10^{-9} \\ -4.03001181 \times 10^{-1} \end{pmatrix}.$$

2.4 Simple/Stable Reduction

There are other choices for the above second step in Section 2.2.2, such as the one resulting in the augmented (quasi-definite) system [50, 46].

In our approach we present a different type of second elimination step. We assume that we have the special structure $A = \begin{pmatrix} B & E \end{pmatrix}$ (perhaps obtained by permuting rows and columns), where B is $m \times m$, nonsingular and not ill-conditioned, and it is inexpensive to solve the corresponding linear system $Bu = d$, i.e. a factorization $B = LU$ can be found with both L and U triangular and sparse. For example, the best choice is $B = I$ obtained when x includes a full set of slack variables. Though it is desirable for B to be well-conditioned, there is no need for B to be a feasible basis matrix.

We partition the diagonal matrices Z, X using the vectors $z = \begin{pmatrix} z_m \\ z_v \end{pmatrix}, \quad x = \begin{pmatrix} x_m \\ x_v \end{pmatrix}$ with lengths m and $v = n - m$. With K given in (2.8), we define the matrices F_s, P_s with

$$\begin{aligned} F_s : \;=\; P_s K &= \begin{pmatrix} I_n & 0 & 0 & 0 \\ 0 & B^{-1} & 0 & 0 \\ 0 & -Z_m B^{-1} & I_m & 0 \\ 0 & 0 & 0 & I_v \end{pmatrix} \begin{pmatrix} 0 & 0 & A^T & I_n \\ B & E & 0 & 0 \\ Z_m & 0 & -X_m B^T & 0 \\ 0 & Z_v & -X_v E^T & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & A^T & I_n \\ I & B^{-1}E & 0 & 0 \\ 0 & \boxed{\begin{matrix} -Z_m B^{-1} E & -X_m B^T \\ Z_v & -X_v E^T \end{matrix}} & 0 \\ 0 & & 0 \end{pmatrix}. \end{aligned} \tag{2.16}$$

The right-hand side becomes

$$\begin{aligned} -P_s P_Z \begin{pmatrix} A^T y + z - c \\ Ax - b \\ ZXe - \mu e \end{pmatrix} \;&=\; -P_s \begin{pmatrix} r_d \\ r_p \\ -X_m (r_d)_m + Z_m X_m e - \mu e \\ -X_v (r_d)_v + Z_v X_v e - \mu e \end{pmatrix} \\ &= \begin{pmatrix} -r_d \\ -B^{-1} r_p \\ Z_m B^{-1} r_p + X_m (r_d)_m - Z_m X_m e + \mu e \\ X_v (r_d)_v - Z_v X_v e + \mu e \end{pmatrix}. \end{aligned} \tag{2.17}$$

Our algorithm uses the last two rows to solve for Δx_v, Δy. We then use the second row to backsolve for Δx_m and then the first row to backsolve for Δz. The matrix B^{-1} is never evaluated, but rather the required operation is performed using a system solve. Therefore, we require this operation to be both efficient and stable. Moreover, if we started with exact dual feasibility and we find the steplength $\alpha > 0$ that maintains positivity for x, z, then we can update $y \leftarrow y + \alpha \Delta y$ first, and then set $z = c - A^T y$; thus we maintain exact dual feasibility (up to the accuracy of the matrix multiplication and vector subtraction). There is no reason to evaluate and carry the residual to the next iteration. This works for the normal equations backsolve as well. But, if we start with exact feasibility for the primal as well, we can also update $x_v \leftarrow x_v + \alpha \Delta x_v$ and then

solve $Bx_m = b - Ex_v$. Thus we guarantee stable primal feasibility as well (up to the accuracy in the matrix vector multiplications and additions, and the system solve for x_m). This is discussed further at the end of Section 2.6.

The matrix derived in (2.16) is generally better conditioned than the one from the normal equations system (2.9) in the sense that, under nondegeneracy assumptions, the condition number is bounded at the solution. We do not change a well-posed problem into an ill-posed one. The result proved in Proposition 2.5 shows the advantages of using this *Stable Reduction*.

2.5 Condition Number Analysis

Proposition 2.5 *Let F_n and F_s be the matrices defined in (2.9) and (2.16). Then, the condition number of F_n diverges to infinity if $x(\mu)_i/z(\mu)_i$ diverges to infinity, for some i, as μ converges to 0. The condition number of F_s is uniformly bounded if there exists a unique primal-dual solution of problems (1.1) and (1.2).*

Proof. Note that

$$F_n^T F_n = \begin{pmatrix} I_n & -Z^{-1}XA^T & 0 \\ -AXZ^{-1} & (AA^T + (AZ^{-1}XA^T)^2 + AZ^{-2}X^2A^T) & A \\ 0 & A^T & I_n \end{pmatrix}. \tag{2.18}$$

We now see, using interlacing of eigenvalues, that this matrix becomes increasingly ill-conditioned. Let $D = Z^{-1}X$. Then the nonzero eigenvalue of $D_{ii}^2 A_{:,i}(A_{:,i})^T$ diverges to infinity, as μ converges to 0. Therefore the largest eigenvalue of the matrix in the middle block $AD^2A^T = \sum_{i=1}^n D_{ii}^2 A_{:,i}(A_{:,i})^T$ must diverge to infinity, i.e. the largest eigenvalue of $F_n^T F_n$ diverges to infinity. Since the smallest eigenvalue cannot exceed 1, this implies that the condition number of $F_n^T F_n$ diverges to infinity, as $\mu \to 0$ and $x(\mu)_i/z(\mu)_i$ diverges to infinity, for some i.

On the other hand, the condition number of F_s is uniformly bounded. This follows from the fact that the submatrix within the box in F_s (2.16) is exactly F'_μ in (2.23). As shown in Theorem 2.8 below, F'_μ is nonsingular at the solution, i.e. F'_0 is nonsingular. Nonsingularity of F_s at $\mu = 0$ now follows from the observation that the two backsolve steps are stable. ■

Remark 2.6 *We can observe that the condition number of the matrix $F_n^T F_n$ is greater than the largest eigenvalue of the block $AZ^{-2}X^2A^T$; equivalently, $\frac{1}{\mathrm{cond}(F_n^T F_n)}$ is smaller than the reciprocal of this largest eigenvalue. With the assumption that x and z stay in a certain neighbourhood of the central path, we know that $\min_i(z_i/x_i)$ is $O(\mu)$. Thus the reciprocal of the condition number of F_n is $O(\mu)$.*

2.6 The Stable Linearization

The stable reduction step above corresponds to the following linearization approach. Recall the primal LP

$$\begin{array}{cccc} & p^* = & \min & c^T x \\ (LP) & & \text{subject to} & Ax = b \\ & & & x \geq 0. \end{array} \tag{2.19}$$

An essential preprocessing step is to find a (hopefully sparse) representation of the null space of A as the range of a matrix N, i.e. given an initial solution $\hat{x}$, we get

$$A\hat{x} = b \Rightarrow \boxed{Ax = b \quad \text{if and only if} \quad x = \hat{x} + Nv, \text{ for some } v \in \Re^{n-m}}.$$

For our method to be efficient, we would like both matrices A, N to be sparse. More precisely, since we use an iterative method, we need both matrix vector multiplications Ax and Nv to be inexpensive. If the original problem is in symmetric form, i.e. if the constraint is of the type

$$Ex \leq b, \quad E \in \Re^{m \times (n-m)},$$

(applications for this form abound, e.g. the diet problem and minimum cost production problem; see e.g. [45, Chap. 16][46]) then we can add slack variables and get $A = (\, I_m \quad E \,), N = \begin{pmatrix} -E \\ I_{n-m} \end{pmatrix}$. More generally, in this paper we assume that

$$A = (\, B \quad E \,), \quad N = \begin{pmatrix} -B^{-1}E \\ I_{n-m} \end{pmatrix}, \tag{2.20}$$

where E is sparse and the linear system $Bv = d$ is nonsingular, well-conditioned and inexpensive to solve. (For example, B is block diagonal or triangular. Surprisingly, this structure holds for most of the NETLIB test set problems. See the comments and the Tables 4.8– 4.10 in Section 4.2.)

We can now substitute for both z, x and eliminate the first two (linear) blocks of equations in the optimality conditions (2.3). We obtain the following **single** block of equations for optimality. By abuse of notation, we keep the symbol F for the nonlinear operator. The meaning is clear from the context.

Theorem 2.7 *Suppose that $A\hat{x} = b$ and the range of N equals the nullspace of A. Also, suppose that $x = \hat{x} + Nv \geq 0$ and $z = c - A^T y \geq 0$. Then the primal-dual variables x, y, z are optimal for (**LP**),(**DLP**) if and only if they satisfy the single bilinear optimality equation*

$$F(v, y) := \text{Diag}\,(c - A^T y)\text{Diag}\,(\hat{x} + Nv)e = 0. \tag{2.21}$$

■

This leads to the single perturbed optimality conditions that we use for our primal-dual method,

$$F_\mu(v,y) := \text{Diag}\,(c - A^T y)\text{Diag}\,(\hat{x} + Nv)e - \mu e = 0. \tag{2.22}$$

This is a nonlinear (bilinear) system. The linearization (or Newton equation) for the search direction $\Delta s := \begin{pmatrix} \Delta v \\ \Delta y \end{pmatrix}$ is

$$F'_\mu(v,y)\Delta s \;\; = \;\; -F_\mu(v,y), \tag{2.23}$$

where the Jacobian $F'_\mu(v,y)$ is the matrix

$$\begin{array}{rcl} J: & = & F'_\mu(v,y) \\ & = & (\,\text{Diag}\,(c - A^T y)N \quad -\text{Diag}\,(\hat{x} + Nv)A^T\,) \\ & = & (\,ZN \quad -XA^T\,). \end{array} \tag{2.24}$$

Therefore, system (2.23) becomes

$$ZN\Delta v - XA^T\Delta y \;\; = \;\; -F_\mu(v,y). \tag{2.25}$$

We note that the first part of the system (2.25) is usually the large part since it has $n - m$ variables Δv. However, this part is inexpensive to evaluate if the matrix E is sparse and the system $Bu = d$ is inexpensive to solve. The second part is usually the small part since it only has m variables Δy. This latter part is the size of the normal equations system that arises in the standard approaches for LP.

Note that algorithms that use reduced linearized systems of this size do exist, e.g. [45, Chap. 19] discusses the *quasi-definite* system of size $n \times n$. These larger systems can be more efficient in the sparse case. In particular, the distinct division into two sets of (almost orthogonal) columns can be exploited using projection and multifrontal methods, e.g. [9, 22, 32, 33, 30]. This allows for parallel implementations that do the QR factorizations for the preconditioning steps.

Under standard assumptions, the above system (2.25) has a unique solution at each point (v,y) that corresponds to a strictly feasible primal-dual pair x, z. In addition, we now show nonsingularity of the Jacobian matrix at optimality, i.e. it does not necessarily get ill-conditioned as μ approaches 0.

Theorem 2.8 *Consider the primal-dual pair (**LP**),(**DLP**). Suppose that A is onto (full rank), the range of N is the null space of A, N has full column rank, and (x,y,z) is the* unique *primal-dual optimal solution. Then the matrix J of the linear system $J\Delta s = -F_\mu$ (2.23) is nonsingular.*

Proof. Suppose that $J\Delta s = 0$. We need to show that $\Delta s = (\Delta v, \Delta y) = 0$.

Let $\mathcal{B}$ and $\mathcal{N}$ denote the set of indices j such that $x_j = \hat{x}_j + (Nv)_j > 0$ and set of indices i such that $z_i = c_i - (A^T y)_i > 0$, respectively. Under the uniqueness (nondegeneracy) and full rank assumptions, we get $\mathcal{B} \bigcup \mathcal{N} = \{1, \ldots, n\}$, $\mathcal{B} \bigcap \mathcal{N} = \emptyset$, and the cardinalities $|\mathcal{B}| = m$,

$|\mathcal{N}| = n - m$. Moreover, the submatrix $A_{\mathcal{B}}$, formed from the columns of A with indices in $\mathcal{B}$, is nonsingular.

By our assumption and (2.25), we have

$$(J\Delta s)_k = (c - A^T y)_k (N\Delta v)_k - (\hat{x} + Nv)_k (A^T \Delta y)_k = 0, \quad \forall k.$$

From the definitions of $\mathcal{B}, \mathcal{N}$ and complementary slackness, this implies that

$$\begin{array}{c} c_j - (A^T y)_j = 0, \quad \hat{x}_j + (Nv)_j > 0, \quad (A^T \Delta y)_j = 0, \quad \forall j \in \mathcal{B}, \\ c_i - (A^T y)_i > 0, \quad \hat{x}_i + (Nv)_i = 0, \quad (N\Delta v)_i = 0, \quad \forall i \in \mathcal{N}. \end{array} \tag{2.26}$$

The first line of (2.26) implies $A_{\mathcal{B}}^T \Delta y = 0$, i.e. we obtain $\Delta y = 0$.

It remains to show that $\Delta v = 0$. From the definition of N we have $AN = 0$. Therefore, (2.26) implies

$$\begin{array}{rcl} 0 & = & (A_{\mathcal{B}} \quad A_{\mathcal{N}}) \begin{pmatrix} (N\Delta v)_{\mathcal{B}} \\ (N\Delta v)_{\mathcal{N}} \end{pmatrix} \\ & = & A_{\mathcal{B}} (N\Delta v)_{\mathcal{B}} + A_{\mathcal{N}} (N\Delta v)_{\mathcal{N}} \\ & = & A_{\mathcal{B}} (N\Delta v)_{\mathcal{B}}. \end{array}$$

By (2.26) and the nonsingularity of $A_{\mathcal{B}}$, we get

$$N\Delta v = 0.$$

Now, full rank of N implies $\Delta v = 0$.

(An alternative proof follows using (2.16). We can see, after permutations if needed, that both K and P_s are nonsingular matrices.) ■

We use equation (2.22) and the linearization (2.25) to develop our primal-dual algorithm. This algorithm is presented and described in the next section.

3 Primal-Dual Algorithm

The algorithm we use follows the primal-dual interior-point framework, see e.g. the books [45],[50, P. 198]. That is, we use Newton's method applied to the perturbed system of optimality conditions with damped step lengths for maintaining nonnegativity (not necessarily positivity) constraints. Our approach differs in that we eliminate, in advance, the primal and dual linear feasibility equations. (Within an infeasible approach, they get eliminated completely only after a steplength of 1 and stay eliminated.) The search direction is found first using a direct factorization in (2.23), and second using a preconditioned conjugate-gradient-type method, LSQR, due to Paige and Saunders [39]. These are applied to the last two rows of (2.16)-(2.17). This contrasts with popular approaches that find the search directions by using direct factorization

methods on the normal equations system. In addition, we use a change to a pure Newton step, i.e. we use affine scaling (the perturbation parameter $\mu = 0$) and we do not backtrack to preserve positivity of z, x once we have found (or estimated) the region of quadratic convergence of Newton's method. Therefore, the algorithm mixes interior and exterior ideas. We also include the identification of zero values for the primal variable x and eliminate the corresponding indices; thus reducing the dimension of the original problem. We call this a *purification step*.

Only indices corresponding to the matrix E are eliminated so that we maintain the $(B \quad E)$ structure. The procedures are explained in more detail in the following sections.

3.1 Initialization and Preprocessing

The preprocessing involves finding B to satisfy the structure in (2.20) with B mostly upper triangular and sparse. However, in some cases we can start the algorithm with a feasible approach i.e. we have initial data $\hat{x}, v_o, y_o$ such that

$$A\hat{x} = b;\ x_o = \hat{x} + Nv_o > 0;\ z_o = c - A^T y_o > 0.$$

The existence of such initial data cannot be assumed in general because finding a feasible solution is just as difficult as solving the problem to optimality. However, special structure can provide this initialization, e.g. suppose that both E, b (and so A) are nonnegative elementwise. Then, with $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, we can set $x_2 = b - Ex_1 > 0$, for sufficiently small $x_1 > 0$, and $v_o = 0$. Similarly, we can choose $z = c - A^T y_o > 0$ for sufficiently negative y_0.

3.2 Preconditioning Techniques

Recall that $Z := Z(y) = \text{Diag}\,(c - A^T y),\ X := X(v) = \text{Diag}\,(\hat{x} + Nv)$, and the Jacobian of F_μ (equation (2.24)) is

$$J := F'_\mu(v, y) = (ZN \quad -XA^T). \tag{3.1}$$

Since we are interested in using a conjugate-gradient-type method for solving the linear system (2.23), we need efficient preconditioners. For a preconditioner we mean a *simple* nonsingular matrix M such that JM^{-1} is well conditioned. To solve system (2.23), we can solve the better conditioned systems $JM^{-1}\Delta q = -F_\mu$ and $M\Delta s = \Delta q$. It is clear that the best condition for JM^{-1} is obtained when the matrix M is the inverse of J. We look for a matrix M such that $M^T M$ approximates $J^T J$.

We use the package LSQR [39], which implicitly solves the normal equations $J^T J\Delta s = -J^T F'_\mu$. Two possible choices for the preconditioning matrix M are: the square root of the diagonal of $J^T J$; and the partial Cholesky factorization of the diagonal blocks of $J^T J$. In the following we describe these approaches. Since our system is nonsymmetric, other choices would be, e.g. quasi-minimal residual (QMR) algorithms [20, 21]. However, preconditioning for these algorithms is more difficult, see e.g. [6, 7].

3.2.1 Optimal Diagonal Column Preconditioning

We begin with the simplest of the preconditioners. For any given square matrix K let us denote $\omega(K) = \frac{\text{trace}\,(K)/n}{\det(K)^{1/n}}$. Let $M = \arg\min \omega((JD)^T(JD))$ over all positive diagonal matrices D. In [16, Prop. 2.1(v)] it was shown that $M_{jj} = 1/\|J_{:j}\|$, the j-th column norm. This matrix has been identified as a successful preconditioner (see [24, Sect. 10.5], [44]) since ω is a measure of the condition number, in the sense that it is bounded above and below by a constant times the standard condition number (ratio of largest and smallest singular values).

3.2.2 Partial (Block) Cholesky Preconditioning

From (3.1) we obtain that

$$J^T J = \begin{pmatrix} N^T Z^2 N & -N^T Z X A^T \\ -AXZN & AX^2A^T \end{pmatrix}.$$

Suppose that z, x lies near the central path, i.e. $ZX \cong \mu I$ (approximately equal). Then the off diagonal terms of $J^T J$ are approximately 0, since $AN = 0$ by definition of N, and XZ is small when μ is small. In this case, block (partial) Cholesky preconditioning is extremely powerful.

We now look at finding a partial Cholesky factorization of $J^T J$ by finding the factorizations of the two diagonal blocks. We can do this using the Q-less QR factorization, i.e. suppose that $Q_Z R_Z = ZN$, $Q_X R_X = XA^T$ represents the QR factorizations with both R_Z and R_X square matrices (using the Q-less efficient form, where Q_Z, Q_R are not stored or formed explicitly). Then

$$R_Z^T R_Z = N^T Z^2 N, \quad R_X^T R_X = AX^2A^T. \tag{3.2}$$

We can now choose the approximate factorization

$$J^T J \cong M^T M, \quad M = \begin{pmatrix} R_Z & 0 \\ 0 & R_X \end{pmatrix}.$$

We should also mention that to calculate this preconditioner is expensive. The expense is comparable to the Cholesky factorization of the normal equation $AZ^{-1}XA^T$. Therefore, we tested both a complete and an incomplete Cholesky preconditioner (denoted IC) for the diagonal blocks.

3.3 Change to Pure Newton Step Technique

Let us assume that the Jacobian of the function F in (2.1) defining the optimality conditions is nonsingular at the solution. Then, the problem has unique primal and dual solutions, $s^* = (\,x^*, y^*, z^*\,)$. Therefore, from the standard theory for Newton's method, there is a neighbourhood of the solution s^* of quadratic convergence and, once in this neighbourhood, we can safely apply affine scaling with step lengths of one without backtracking to maintain positivity of x or z.

To estimate the guaranteed convergence area of the optimal solution, we need to use a theorem due to Kantorovich [28]. We use the form in [15, Theorem 5.3.1]. Let $\mathcal{N}(x,r)$ denote the neighbourhood of x with radius r, and $\mathrm{Lip}_\gamma(\mathcal{N}(x,r))$ denote Lipschitz continuity with constant γ in the neighbourhood.

Theorem 3.1 (Kantorovich) *Suppose $r > 0$, $s_0 \in \Re^n$, $F : \Re^n \to \Re^n$, and that F is continuously differentiable in $\mathcal{N}(s_0, r)$ with $J(s_0)$ nonsingular. Assume for a vector norm and the induced operator norm that $J \in Lip_\gamma(\mathcal{N}(s_0, r))$ for some Lipschitz constant γ, and that constants β, $\eta \geq 0$ exist such that*

$$\|J(s_0)^{-1}\| \leq \beta, \quad \|J(s_0)^{-1}F(s_0)\| \leq \eta.$$

Define $\alpha = \beta\gamma\eta$. If $\alpha \leq \frac{1}{2}$ and $r \geq r_0 := (1 - \sqrt{1-2\alpha})/(\beta\gamma)$, then the sequence $\{s_k\}$ produced by

$$s_{k+1} = s_k - J(s_k)^{-1}F(s_k), \quad k = 0, 1, \ldots,$$

is well defined and converges to s_*, a unique zero of F in the closure of $\mathcal{N}(s_0, r_0)$. If $\alpha < \frac{1}{2}$, then s_* is the unique zero of F in $\mathcal{N}(s_0, r_1)$, where $r_1 := \min[r, (1 + \sqrt{1-2\alpha})/(\beta\gamma)]$ and

$$\|s_k - s_*\| \leq (2\alpha)^{2^k}\frac{\eta}{\alpha}, \quad k = 0, 1, \ldots.$$

■

We follow the notation in Dennis and Schnabel [15] and find the Lipschitz constant used to determine the region of quadratic convergence.

Lemma 3.2 *The Jacobian $F'(v,y) = (\,\mathrm{Diag}\,(c - A^T y)N \quad -\mathrm{Diag}\,(\hat{x} + Nv)A^T\,)$ in (3.1) is Lipschitz continuous with constant*

$$\gamma = \sigma_{\max}(F' - F'(0)) \leq \sqrt{2}\|A\|\|N\|, \tag{3.3}$$

where $\sigma_{\max}(F' - F'(0))$ is the largest singular value of the linear transformation $G(v,y) := F'(v,y) - F'(0) : \Re^n \to \Re^{n\times n}$.

Proof. For each $s = (v,y) \in \Re^n$ we get the matrix $F'(s) \in \Re^{n\times n}$. This mapping is denoted by the affine transformation $F' : \Re^n \to \Re^{n\times n}$. Therefore $G(s) := F'(s) - F'(0)$ is a linear transformation. The largest singular value of the matrix representation is denoted $\sigma_{\max} := \sigma_{\max}(G)$. This satisfies $\|F'(s) - F'(\bar{s})\| = \|G(s - \bar{s})\| \leq \sigma_{\max}\|s - \bar{s}\|$. Hence by setting $s = 0$ and $\bar{s}$ to be the singular vector corresponding to the largest singular value, we conclude $\gamma = \sigma_{\max}$.

Now let $\Delta s = \begin{pmatrix} \Delta v \\ \Delta y \end{pmatrix}$. Since

$$\|F'(s) - F'(\bar{s})\| \quad = \quad \max \frac{\|(F'(s) - F'(\bar{s}))\Delta s\|}{\|\Delta s\|}$$

$$
\begin{aligned}
&= \max \frac{\|\mathrm{Diag}\,(A^T(y-\bar{y}))N\Delta v - \mathrm{Diag}\,(A^T\Delta y)N(v-\bar{v})\|}{\|\Delta s\|} \\
&\leq \max \frac{\|A^T(y-\bar{y})\|\|N\Delta v\| + \|A^T\Delta y\|\|N(v-\bar{v})\|}{\|\Delta s\|} \\
&\leq \|A\|\|N\|\|y-\bar{y}\| + \|A\|\|N\|\|v-\bar{v}\| \\
&\leq \sqrt{2}\|A\|\|N\|\|s-\bar{s}\|,
\end{aligned}
$$

a Lipschitz constant is $\gamma = \sqrt{2}\|A\|\|N\|$. ■

Observe that the Lipschitz constant depends on the representation matrix N that we consider. In particular, N can be chosen so that its columns are orthonormal and $\|N\Delta v\|_= \|\Delta v\|$ and $\|N(v-\bar{v})\| = \|v-\bar{v}\|$. In this case, the Lipschitz constant $\gamma \leq \sqrt{2}\|A\|$.

We can evaluate the largest singular value $\sigma_{\max}$ in the above Theorem 3.1 as follows. Consider the linear transformation $\mathcal{L} : \mathbb{R}^n \mapsto \mathbb{R}^{n^2}$ defined by

$$
\mathcal{L}\begin{pmatrix} v \\ y \end{pmatrix} := \mathrm{vec}\ (\ \mathrm{Diag}\,(A^T y)N \quad \mathrm{Diag}\,(Nv)A^T\ ),
$$

where $\mathrm{vec}\,(M)$ denotes the vector formed columnwise from the matrix M. The inverse of vec is denoted Mat . Let $w \in \mathbb{R}^{n^2}$. The inner-product

$$
\begin{aligned}
\left\langle \mathcal{L}\begin{pmatrix} v \\ y \end{pmatrix}, w \right\rangle &= \langle \mathrm{vec}\ (\ \mathrm{Diag}\,(A^T y)N \quad \mathrm{Diag}\,(Nv)A^T\ ), w\rangle \\
&= \left\langle \begin{pmatrix} v \\ y \end{pmatrix}, \begin{pmatrix} N^T \mathrm{diag}\,(A^T W_2^T) \\ A\,\mathrm{diag}\,(NW_1^T) \end{pmatrix} \right\rangle,
\end{aligned}
$$

where W_1 is the first $n-m$ columns of $\mathrm{Mat}\,(w)$ and W_2 is the remaining m columns of $\mathrm{Mat}\,(w)$. Therefore, the adjoint operator of $\mathcal{L}$ is

$$
\mathcal{L}^*(w) = \begin{pmatrix} N^T \mathrm{diag}\,(A^T W_2^T) \\ A\,\mathrm{diag}\,(NW_1^T) \end{pmatrix}.
$$

We can use a few iterations of the power method to approximate efficiently the largest eigenvalue of $\mathcal{L}^*\mathcal{L}$ (which is the equivalent to the square of the largest singular value of $\mathcal{L}$). This can be done without forming the matrix representation of $\mathcal{L}$.

We also need to estimate β, the bound on the norm of the inverse of the Jacobian at the current $s = (v, y)$, i.e.

$$
\beta \geq \|J^{-1}\| = \frac{1}{\sigma_{\min}(J)}. \tag{3.4}
$$

Finally, to estimate η, we note that

$$
\|J^{-1}F_0(v,y)\| = \|J^{-1}(-ZXe)\| \leq \eta. \tag{3.5}
$$

The vector $J^{-1}(-ZXe)$ is the affine scaling direction and is available within the predictor-corrector approach that we use.

We now have the following heuristic for our change to a pure Newton step technique.

Theorem 3.3 *Suppose that the constants γ, β, η in Theorem 3.1 satisfy* (3.3)-(3.4)-(3.5)*, respectively. And, suppose that $s_0 = (v_0, y_0)$ and $\alpha = \gamma\beta\eta < \frac{1}{2}$. Then the undamped Newton sequence s_k generated by $s_{k+1} = s_k - J(s_k)^{-1}F(s_k)$ converges to s^*, the unique zero of F in the neighbourhood $\mathcal{N}(s_0, r_1)$.* ■

Remark 3.4 *Theorem 3.3 guarantees convergence of the affine scaling direction to a solution of $F(s) = 0$ without backtracking. But, it does not guarantee convergence to a solution with x, z nonnegative. Nonetheless, all our numerical tests successfully found nonnegative solutions.*

3.4 Purify Step

Purifying refers to detecting variables that are zero at optimality. This is equivalent to identifying active constraints, e.g. [10, 11, 12]. We use the Tapia indicators [18] to detect the x variables going to zero. (See also [36, 1].) This is more difficult than the change to a pure Newton step, as variables can increase and decrease while converging to 0, see e.g. [23].

Our tests were divided into two cases. Our *infeasible code* has a starting point that satisfies strict positivity, but primal-dual feasibility $Ax = b, A^T y + z = c$ may fail. For this case, once we identify a variable x_j converging to zero, we remove the corresponding column in A and components in c, z. The infeasibility at the next iteration stays small. To maintain the $(\,B \quad E\,)$ structure of our data matrix A, we limit our choice of dropping variables to those associated with E. In the case of our *feasible code* (our starting point satisfies positivity as well as both primal and dual feasibility), we have more involved book-keeping so that we maintain feasibility after dropping variables with small positive values.

4 Numerical Tests

Our numerical tests use the well known NETLIB LP data library as well as randomly generated data. We compare our algorithm with the well known MATLAB based linear programming solver LIPSOL [54], www.caam.rice.edu/~zhang/lipsol/. (We use the same preprocessing as LIPSOL: delete fixed variables; delete zero rows and columns; ensure that A is structurally *full rank*; shift nonzero lower bounds; find upper bounds.)

Our randomly generated problems use data A, b, c, with a known optimal basis in A and optimal values x, y, and z. For the infeasible code tests, we used the same starting point strategy given in LIPSOL. For the feasible code tests we applied one Newton step from the optimal point with a large positive μ, in order to maintain feasibility of the starting point. In addition, we

data	m	n	nnz(E)	cond($A_{\mathcal{B}}$)	cond(J)	***NEQ***		Stable direct	
						D_time	its	D_Time	its
1	100	200	1233	51295	32584	0.03	$*$	0.06	6
2	200	400	2526	354937	268805	0.09	6	0.49	6
3	200	400	4358	63955	185503	0.10	$*$	0.58	6
4	400	800	5121	14261771	2864905	0.61	$*$	3.66	6
5	400	800	8939	459727	256269	0.64	6	4.43	6
6	800	1600	10332	11311945	5730600	5.02	6	26.43	6
7	800	1600	18135	4751747	1608389	5.11	$*$	33.10	6

Table 4.1: nnz(E) - number of nonzeros in E; cond($\cdot$) - condition number; $A_{\mathcal{B}}$ optimal basis matrix, $J = (ZN \;\; -XA^T)$ at optimum, see (3.1); D_time - avg. time per iteration for search direction, in sec.; its - iteration number of interior point methods. * denotes ***NEQ*** stalls at relative gap 10^{-11}.

ensure that the Jacobian of the optimality conditions at the optimum is nonsingular (so the optimal x, y, z are unique) and its condition number is not *large*, since we want to illustrate how the stable system takes advantage of well-conditioned, nondegenerate, problems. The iteration is stopped when the relative duality gap (including the relative infeasibility) is less than 10^{-12}. The computations were done in MATLAB 6.5 on a 733MHz Pentium 3 running Windows 2000 with 256MB RAM.

To find the search direction, we use either a direct or iterative method to solve $J\Delta s = -F_\mu$. The direct method uses [L,U,P,Q]=lu($\cdot$) in MATLAB to find LU factors of J. The permutations P,Q exploit the sparsity of J. (Note that using lu($\cdot$) is generally slower than using \ with a single right-hand side, but we have two right-hand sides (for the predictor and corrector steps) and use the factors twice.) The iterative method uses an inexact Newton approach. The linear system is solved approximately using LSQR [39] with different preconditioners. We use adaptive tolerance settings for LSQR: $atol = \max(10^{-13}, 10^{-10}\mu), btol = \max(10^{-10}, 10^{-10}\mu)$. Both direct and iterative approaches share the same interior-point framework and include the change to a pure Newton and purify steps. They differ only in the method used for computing the search direction.

The normal equation, ***NEQ*** , approach uses chol($\cdot$) in MATLAB to find a Cholesky factorization of $AZ^{-1}XA^T$. It then uses the Cholesky factor with the MATLAB \ (backslash) in both the predictor and corrector step. (We note that using "chol($\cdot$)" is generally three times slower than using \ (backslash) directly on ***NEQ*** .) The ***NEQ*** approach can solve many of the random generated problems to the required accuracy. However, if we set the stopping tolerance to 10^{-15}, we do encounter quite a few examples where ***NEQ*** stalls with relative gap approximately 10^{-11}, while the stable system has no problem reaching the desired accuracy.

data set	LSQR with IC for diagonal blocks				LSQR with Diag			
	D_Time	its	L_its	Pre_time	D_Time	its	L_its	Pre_time
1	0.15	6	37	0.06	0.41	6	556	0.01
2	3.42	6	343	0.28	2.24	6	1569	0.00
3	2.11	6	164	0.32	3.18	6	1595	0.00
4	NA	Stalling	NA	NA	13.37	6	4576	0.01
5	NA	Stalling	NA	NA	21.58	6	4207	0.01
6	NA	Stalling	NA	NA	90.24	6	9239	0.02
7	NA	Stalling	NA	NA	128.67	6	8254	0.02

Table 4.2: Same data sets as in Table 4.1; two different preconditioners (incomplete Cholesky with drop tolerance 0.001 and diagonal); D_time - average time for search direction; its - iteration number of interior point methods; L_its - average number of LSQR iterations per major iteration; Pre_time - average time for preconditioner; Stalling - LSQR cannot converge due to poor preconditioning.

data set	LSQR with block Chol. Precond.			
	D_Time	its	L_its	Pre_time
1	0.09	6	4	0.07
2	0.57	6	5	0.48
3	0.68	6	5	0.58
4	5.55	6	6	5.16
5	6.87	6	6	6.45
6	43.28	6	5	41.85
7	54.80	6	5	53.35

Table 4.3: Same data sets as in Table 4.1; LSQR with Block Cholesky preconditioner. Notation is the same as Table 4.2.

The tests in Tables 4.1-4.2-4.3 are done without the change to a pure Newton step and purification techniques. The stable method with the direct solver and also with the diagonal preconditioner consistently obtains high accuracy optimal solutions. The stable method is not competitive in terms of time compared to the ***NEQ*** approach for this test set. One possible reason is that the condition numbers of J, the Jacobian at the optimum, and of the basis matrix $A_{\mathcal{B}}$, are still too large for the iterative method to be effective. We provide another set of numerical tests based on well conditioned $A_{\mathcal{B}}$ in the following subsection.

We also performed many tests with the change to a pure Newton step. Using our test for α in Theorem 3.3 with the inexpensive bound for γ, we can usually detect the guaranteed convergence region at $\mu = 10^{-6}$ or with the relative gap tolerance at 10^{-4} or 10^{-5}. We also encountered a few examples where the change begins as early as $\mu = 10^{-4}$ and some examples where the change begins as late as $\mu = 10^{-8}$. After the test is satisfied, we use an undamped Newton method, i.e. we use the affine scaling direction with step length 1 without limiting x and z to be nonnegative. It usually takes only one iteration to achieve the required accuracy 10^{-12}. This is not a surprise considering the quadratic convergence rate of Newton's method.

If we compare the method without a change to a pure Newton step, then we conclude that the change technique gives an average 1 iteration saving to achieve the desired accuracy. We also encountered several instances where ***NEQ*** did not converge after the change to a pure Newton step, while our stable method had no difficulty. We should mention that ***NEQ*** is not suitable for a pure Newton step because the Jacobian becomes singular. Moreover, a catastrophic error occurs if a z element becomes zero.

We also tested the purification technique. It showed a benefit for the stable direction when n was large compared to m, since we only identify nonbasic variables. (However, deleting variables does not help ***NEQ*** because $AXZ^{-1}A^T$ remains $m \times m$.) The time saving on solving the linear system for the stable direction is cubic in the percentage of variables eliminated, e.g. if half the variables are eliminated, then the time is reduced to $(\frac{1}{2})^3 = \frac{1}{8}$ the original time. The purification technique starts to identify nonbasic variables as early as 6–7 iterations before convergence. It usually identifies most of the nonbasic variables from two to four iterations before convergence. For all our random generated tests, the purification technique successfully identified all the nonbasic variables before the last two iterations.

We should also mention the computation costs. For the change to a pure Newton step, we need to evaluate the smallest singular value of a sparse $n \times n$ matrix to find β, and then solve an $n \times n$ linear system to find the value η (see Theorem 3.3). The cost of finding the smallest singular value is similar to that of solving a system of the same size. Solving this linear system is inexpensive since the matrix J is the same as for the search direction and we already have a factorization.

In the above tests we restricted ourselves to nondegenerate problems. See Figure 4.1 for a comparison on a typical degenerate problem. Note that ***NEQ*** had such difficulties on more than half of our degenerate test problems.

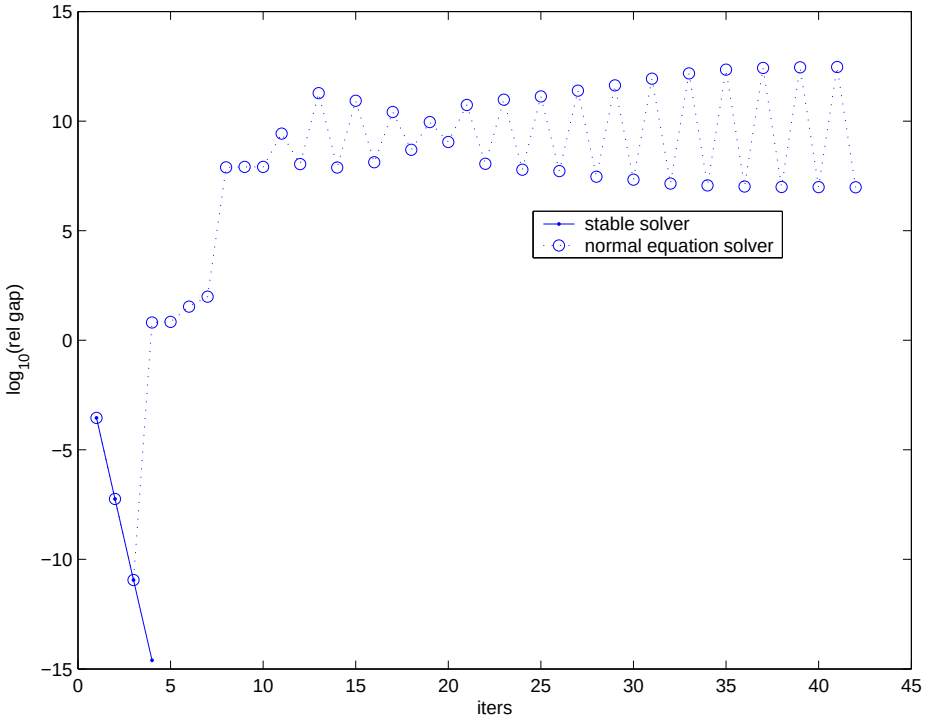

Figure 4.1: Iterations for Degenerate Problem

4.1 Well Conditioned $A_{\mathcal{B}}$

Our previous test examples in Tables 4.1-4.2-4.3 are all sparse with 10 to 20 nonzeros per row. In this section we generate *sparser* problems with about 3-4 nonzeros per row in E but we still maintain nonsingularity of the Jacobian at the optimum. We first fix the indices of a basis $\mathcal{B}$; we choose half of the column indices j so that they satisfy $1 \leq j \leq m$ and the other half satisfy $m+1 \leq j \leq n$. We then add a random diagonal matrix to $A_{\mathcal{B}}$ to obtain a well-conditioned basis matrix and generate two random (sufficiently) positive vectors $x_{\mathcal{B}}$ and $z_{\mathcal{N}}$. We set the optimal $x^* = \begin{pmatrix} x_{\mathcal{B}} \\ x_{\mathcal{N}} \end{pmatrix}$ with $x_{\mathcal{N}} = 0$; and the optimal $z^* = \begin{pmatrix} z_{\mathcal{B}} \\ z_{\mathcal{N}} \end{pmatrix}$, with $z_{\mathcal{B}} = 0$. The data b, c are determined from $b := Ax^*, \quad c := A^T y^* + z^*, \quad y^* \in \Re^m$ random (using MATLAB's "randn").

We now compare the performance of three different solvers for the search direction, namely ***NEQ*** solver, direct linear solver on the stable system, and LSQR on the stable system. In this section, we restrict ourselves to the diagonal preconditioner when we use the LSQR solver. (The computations in this section were done on a Sun-Fire-480R running SunOS 5.8.)

The problems in Table 4.4 all have the same dimensions. To illustrate that our method can

data sets				***NEQ***		Stable Direct		LSQR		
Name	cond($A_{\mathcal{B}}$)	cond(J)	nnz(E)	D_Time	its	D_Time	its	D_Time	its	L_its
nnz2	19	14000	4490	3.75	7	5.89	7	0.19	7	81
nnz4	21	20000	6481	3.68	7	7.38	7	0.27	7	106
nnz8	28	10000	10456	3.68	7	11.91	7	0.42	7	132
nnz16	76	11000	18346	3.69	7	15.50	7	0.92	7	210
nnz32	201	12000	33883	3.75	9	18.43	9	2.29	8	339

Table 4.4: *Sparsity vs Solvers:* cond(·) - (rounded) condition number; D_time - average time for search direction; its - number of iterations; L_its - average number LSQR iterations per major iteration; All data sets have the same dimension, 1000×2000, and have 2 dense columns.

handle sparse problems without additional special techniques, we include two full dense columns (in E). We let the total number of nonzeros increase. The condition numbers are evaluated using the MATLAB "condest" command. The loss in sparsity has essentially no effect on ***NEQ***, since the ADA^T matrix is already dense because of the two dense columns. But we can see the negative effect that the loss of sparsity has on the stable direct solver, since the density in the system (2.24) increases. For these problem instances, using LSQR with the stable system can be up to twenty times faster than the ***NEQ*** solver.

Our second test set in Table 4.5 shows how size affects the three different solvers. The time for the ***NEQ*** solver is proportional to m^3. The stable direct solver is about twice that of ***NEQ***. LSQR is the best among these 3 solvers on these instances. The computational advantage of LSQR becomes more apparent as the dimension grows.

We also use LIPSOL to solve our test problems, see Table 4.6. Our tests use LIPSOL's default settings except that the stopping tolerance is set to 10^{-12}. LIPSOL uses a primal-dual infeasible-interior-point algorithm. We can see that the number of iterations for LIPSOL are in a different range from our tests in Tables 4.4, 4.5 which are usually in the range of 6-8. It can be observed that LIPSOL in general performs better than the ***NEQ*** code we have written. Since LIPSOL has some special code to deal with factorization, while our method just uses the LU (or chol) factorization from MATLAB, it is not unexpected to see the better performance from LIPSOL.

But comparing to the iterative method, we should mention that when the problem size becomes large, the iterative method has an obvious advantage over the direct factorization method. This can be seen clearly from the solution times of problems sz8-sz9-sz10 in Table 4.6 and the corresponding time of LSQR in Table 4.5. When the problem size doubles, the solution time for LIPSOL increases roughly by a factor of 8-10, while the solution time for our iterative method roughly doubles. This is also true for fully sparse problems as mentioned in the caption of Table 4.6.

The iterative solver LSQR does not spend the same amount of time at different stages of an

data sets				***NEQ***		Stable Direct		LSQR	
name	size	cond($A_{\mathcal{B}}$)	cond(J)	D_Time	its	D_Time	its	D_Time	its
sz1	400×800	20	2962	0.29	7	0.42	7	0.07	7
sz2	400×1600	15	2986	0.29	7	0.42	7	0.11	7
sz3	400×3200	13	2358	0.30	7	0.43	7	0.19	7
sz4	800×1600	19	12340	1.91	7	3.05	7	0.13	7
sz5	800×3200	15	15480	1.92	7	3.00	7	0.27	7
sz6	1600×3200	20	53240	16.77	7	51.52	7	0.41	7
sz7	1600×6400	16	56810	16.70	7	51.75	7	0.65	8
sz8	3200×6400	19	218700	240.50	7	573.55	7	0.84	7
sz9	6400×12800	24	$8.9e+5$					2.20	6
sz10	12800×25600	22	$2.4e+5$					4.67	6

Table 4.5: *How problem dimension affects different solvers:* cond($\cdot$) - (rounded) condition number; D_time - average time for search direction; its - number of iterations. All the data sets have 2 dense columns in E. The sparsity for the data sets are similar; without the 2 dense columns, they have about 3 nonzeros per row.

interior point method. To illustrate this, we take the data set in Table 4.4. For each problem we draw the number of LSQR iterations at each iteration; see Figure 4.2.

4.2 NETLIB Set - Ill-conditioned Problems

The NETLIB ***LP*** data set is made up of large, sparse, highly degenerate problems, which result in singular Jacobian matrices at the optimum. These problems are ill-posed in the sense of Hadamard; we used the measure in [38] and found that 71% of the problems have infinite condition number. (See also [29].) In particular, small changes in the data can result in large changes in the optimum x, y, z, see e.g. [4, 5],[3, Pages 9-10], [43, Chapter 8]. Therefore, infeasibility is difficult to detect and, it is not evident what a non-regularized solution of these problems means. Nevertheless, we applied our method to these problems. Though our method solves the problems in the NETLIB data set to high accuracy, our tests show that it is *not* competitive (with regard to cpu times) compared to standard ***LP*** packages such as LIPSOL version 0.60 [54], when applied exclusively to the NETLIB data set. Ill-conditioning of J in our algorithm affects the performance of iterative solvers. Direct factorization is preferable for the NETLIB set.

For general LP problems, we want to find a B that is sparse and easy to factorize in the $(\,B \quad E\,)$ structure. An upper triangular matrix is a good choice. The heuristic we use is to go through the columns of the matrix A and find those columns that only have one nonzero entry. We then permute the columns and rows so that these nonzero entries are on the diagonal of B. (In the case of multiple choices in one row, we picked the one with the largest magnitude.) We

data sets	LIPSOL	
name	D_Time	its
nnz2	0.08	12
nnz4	0.50	14
nnz8	1.69	14
nnz16	2.72	14
nnz32	3.94	13
sz1	0.16	11
sz2	0.15	13
sz3	0.15	14
sz4	0.05	12
sz5	0.03	14
sz6	0.22	15
sz7	0.06	15
sz8	1.55	14
sz9	12.80	15
sz10	126.47	15

Table 4.6: *LIPSOL results* D_time - average time for search direction; its - number of iterations. (We also tested problems sz8,sz9,sz10 with the change two dense columns replaced by two sparse columns, only 6 nonzeros in these new columns. (D_time, iterations) on LIPSOL for these fully sparse problems: $(0.41, 11)$, $(2.81, 11)$, $(43.36, 11)$.)

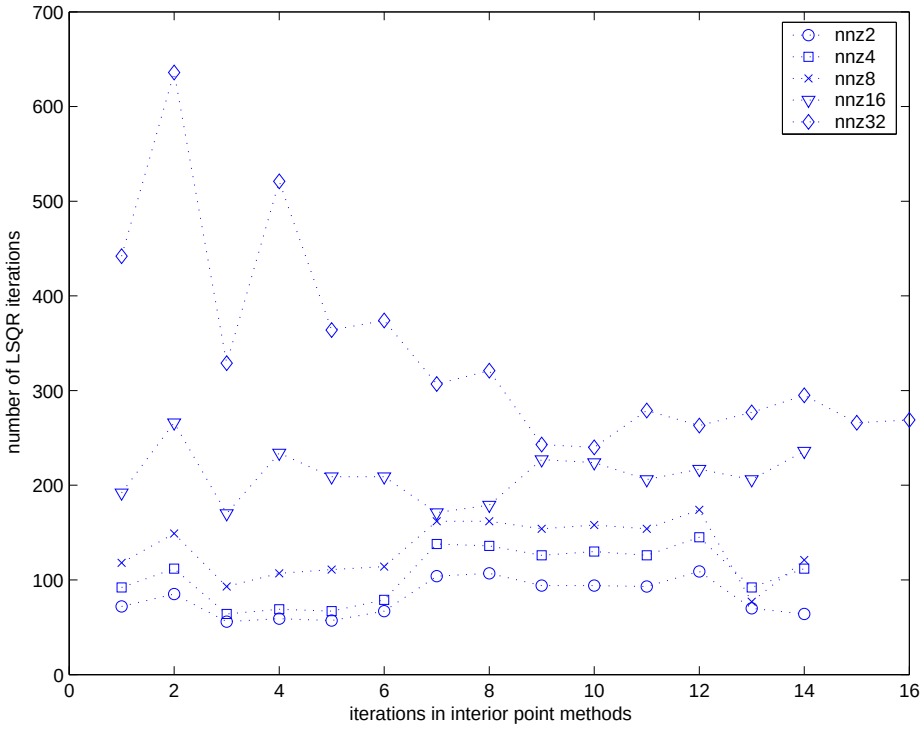

Figure 4.2: LSQR iterations for data set in Table 4.4. Odd-numbered iterations are predictor steps; even-numbered iterations are corrector steps.

remove the corresponding rows and columns, and then repeat the procedure on the remaining submatrix. If this procedure is successful, we end up with an upper triangular matrix B. However, sometimes, we may have a submatrix $\hat{A}$ of A such that no column has one nonzero entry. Usually, such a submatrix $\hat{A}$ is much smaller in size. We use an LU factorization on this small submatrix and find an upper triangular part $\hat{U}$ in the U part of the LU factorization by using the above procedure. The B is then determined by incorporating those columns of $\hat{U}$ after an appropriate permutation. This procedure also results in a useful LU factorization for B. In our tables, we denote the row dimension of the $\hat{A}$ as *no-tri-size of* B. For NETLIB problems, surprisingly, most of them have a zero no-tri-size of B as shown in Tables 4.8–4.10. It is worth noting that some of the NETLIB problems may not have full row rank or the LU factorization on the submatrix $\hat{A}$ may not give an upper triangular U. Thus we may not be able to identify the upper triangular matrix $\hat{U}$. In Tables 4.8–4.10, these problems are marked with a " *" in the column of no-tri-size of B. For these singular problems, our solver may not give a correct answer. (This issue can be resolved by preprocessing to eliminate redundant rows and by a better LU factorization. This is beyond the scope of this paper.) Among these singular

NETLIB problems	Highest Accuracy Attained
bnl2	infeasible
cycle	$9.19e{-}11$
dfl001	infeasible
etamacro	$7.66e{-}11$
fit1p	infeasible
fit2p	infeasible
greenbea	infeasible
grow15	$4.35e{-}10$
grow22	$9.24e{-}10$
grow7	$2.62e{-}10$
kb2	$3.75e{-}12$
pilot87	$1.21e{-}8$
seba	infeasible

Table 4.7: LIPSOL failures with desired tolerance $1e{-}12$; highest accuracy attained by LIPSOL.

problems, "bore3d" and "standgub" have a complete zero row; thus we can easily identify the linearly dependent row in the matrix A and remove it. Our answers for these two problems are accurate.

To make a fair comparison on the errors, we changed the error term in LIPSOL to be the same as ours, which is defined as

$$\text{error} := \frac{|c^T x - b^T y|}{1 + |c^T x|} + \frac{\|r_p\|}{1 + \|b\|} + \frac{\|r_d\|}{1 + \|c\|}. \tag{4.1}$$

We note that LIPSOL can solve all the NETLIB problems to 8 decimal accuracy. In addition, we added the preprocessing step that LIPSOL is using to our code.

We observed improved robustness when using our stable direct factorization method. For example, when the stopping tolerance is set to 12 decimals, LIPSOL could not solve the subset of NETLIB problems in Table 4.7 and, incorrectly, finds that several problems are infeasible. Table 4.7 lists the highest accuracy that LIPSOL can get. (LIPSOL does solve problems *fit1p, fit2p, seba* when the stopping tolerance is set to 10^{-8} and does solve problems *bnl2, dfl001, greenbea* with tolerance 10^{-8} and its own error term.) This illustrates the numerical difficulties that arise for ***NEQ*** based methods when the requested accuracy is more than 10^{-8}. Our stable direct factorization method not only achieved the desired accuracy (except for *capri* with $1.2e{-}12$, *pilot.ja* with $3.7e{-}12$, *pilot* with $6.7e{-}12$) but also exhibited quadratic convergence during the final few iterations on these problems. For complete results on the NETLIB problem, see Tables 4.8–4.10. Further numerical tests appear in the forthcoming [47, 42] and in the recent Masters

thesis [41]. In [41, 42], a different transformation on the NETLIB problems is used to obtain the $(\,I \quad E\,)$ structure. The numerical tests on the NETLIB problems in [41, 42] show that the ill-conditioning negatively affects the performance of the stable algorithm. However, it also observed that much more accurate solutions were obtained by using the stable linearization approach compared to ***NEQ***. Tests for quadratic programs are done in [17].

4.3 No Backtracking

We now present some interesting numerical results under the condition that the interior point method takes a complete step to the boundary without the customary backtracking that guarantees sufficient positivity of the variables x, z. We present the results from the three algorithms: (i) ***NEQ*** with backtracking; (ii) stable system with backtracking; (iii) stable system with no backtracking. Since the ***NEQ*** approach is undefined at the boundary, we cannot include a fourth comparison. No backtracking does not create problems for our stable system, since we do not need the inverse of X or Z.

See Figure 4.3 for a comparison between ***NEQ*** with backtracking and the stable direction with and without backtracking. In this example, the relative gap stopping tolerance for ***NEQ*** is set to 10^{-12}, which is the highest accuracy ***NEQ*** can get for this problem. However, the relative gap stopping tolerances for both of the stable system approaches are set to 10^{-14}. For the first 4 iterations the three approaches are almost indistinguishable, since the backtrack (we backtrack with .9998) is such a small step. However, once the duality gap is small, no backtracking means we are close to taking a complete Newton step so we get a large improvement with the no-backtracking strategy. We reach the desired tolerance in 6 iterations compared to 8 for the stable direction with backtracking. The difference with using backtracking for the stable direction is typical; while stalling for ***NEQ*** occurs for about half our tests.

For many tests, we see that the number of iterations are reduced and the last step behaves just as if the change to a pure Newton step was implemented, i.e. we jump to the stopping tolerance of 14 decimals. This is probably due to the fact that a full step to the boundary is closer to a full Newton step, i.e. this is comparable to implementing the pure Newton step technique. On average, the stable direct method without backtracking results in a 1-2 reduction in the number of iterations.

5 Conclusion

We have presented a robust alternative for interior-point solutions of ***LP***s. We used a preprocessing step to eliminate both the primal and dual (linear) feasibility equations. We then applied an inexact Newton approach to the resulting linear system. We compared this method to the ***NEQ*** approach.

Advantages of our approach include:

problems	LIPSOL			Stable Direct			
Name	D_time	its	error	D_time	its	error	no-tri-size of B
25fv47	0.05	25	1.21e-14	0.94	24	8.7e-15	2
80bau3b	0.14	41	4.38e-14	2.84	49	5.5e-13	0
adlittle	0.01	12	4.13e-14	0.01	12	3.7e-16	2
afiro	0.01	8	3.70e-15	0.00	8	3.5e-16	0
agg	0.03	19	1.06e-13	0.10	19	4.5e-13	0
agg2	0.03	17	1.28e-13	0.19	17	1.4e-15	0
agg3	0.03	17	2.38e-15	0.18	16	1.4e-13	0
bandm	0.01	20	1.77e-14	0.05	17	2.3e-15	0
beaconfd	0.01	13	3.64e-14	0.04	13	3.0e-15	0
blend	0.01	12	8.32e-13	0.01	12	3.4e-15	0
bnl1	0.02	28	2.32e-14	0.37	27	3.0e-14	8
bnl2	0.08	7	2.40e+01	2.01	51	7.3e-13	0
boeing1	0.03	22	1.46e-13	0.14	23	4.7e-15	0
boeing2	0.01	20	1.46e-14	0.03	17	7.9e-13	0
bore3d	0.01	18	9.62e-14	0.03	18	3.3e-14	4*
brandy	0.01	17	8.37e-15	0.04	15	4.2e-13	52
capri	0.02	19	2.76e-13	0.06	20	1.2e-12	0
cycle	0.12	36	9.19e-11	1.98	29	2.5e-13	4
czprob	0.03	36	7.91e-14	1.06	34	7.1e-13	0
d2q06c	0.18	33	1.92e-14	6.21	30	2.1e-13	132*
d6cube	0.11	25	1.23e-15	3.54	14	4.8e-14	404*
degen2	0.03	14	3.62e-13	0.14	13	2.4e-15	97*
degen3	0.25	29	1.22e-13	2.02	17	3.8e-13	159*
dfl001	19.63	17	2.28e+00	46.65	52	1.0e+01	4275*
e226	0.01	22	1.05e-13	0.06	21	3.7e-13	0
etamacro	0.02	45	7.66e-11	0.11	37	7.3e-13	16
fffff800	0.03	27	9.21e-14	0.21	25	4.1e-14	0
finnis	0.02	30	7.40e-13	0.08	27	8.6e-13	0
fit1d	0.04	24	4.18e-13	0.50	18	9.2e-15	0
fit1p	0.30	17	1.75e-05	0.25	16	9.2e-14	0
fit2d	0.43	26	7.05e-13	80.99	23	8.4e-15	0
fit2p	0.68	22	2.35e-07	5.76	23	5.1e-14	0
forplan	0.02	23	1.98e-13	0.09	28	7.9e-13	0
ganges	0.04	19	5.14e-14	0.28	20	9.6e-13	12
gfrd-pnc	0.02	20	3.53e-14	0.1	20	9.9e-15	0

Table 4.8: NETLIB set with LIPSOL and Stable Direct method. D_time - avg. time per iteration for search direction, in sec.; its - iteration number of interior point methods.

problems	LIPSOL			Stable Direct			
Name	D_time	its	error	D_time	its	error	no-tri-size of B
greenbea	0.24	32	6.01e-04	5.68	45	4.6e-13	2
greenbeb	0.15	38	2.01e-13	5.49	37	6.1e-14	2
grow15	0.03	31	4.35e-10	0.86	12	2.4e-13	0
grow22	0.04	25	9.24e-10	2.27	14	4.3e-14	0
grow7	0.02	37	2.62e-10	0.16	12	2.2e-15	0
israel	0.02	23	5.06e-13	0.04	23	9.6e-14	0
kb2	0.01	34	3.75e-12	0.01	16	1.1e-14	0
lotfi	0.01	19	1.51e-15	0.05	17	9.5e-13	0
maros-r7	2.03	15	1.43e-15	14.97	15	1.3e-15	0
maros	0.05	33	5.24e-13	0.59	31	1.1e-13	4
modszk1	0.02	25	3.23e-13	0.22	68	9.8e-13	0
nesm	0.06	35	1.45e-13	2.77	32	7.3e-13	0
perold	0.04	32	5.66e-13	0.71	37	6.4e-13	0
pilot.ja	0.30	33	2.63e-13	1.34	35	3.7e-12	0
pilot	0.07	35	7.72e-13	13.69	42	6.7e-12	0
pilot.we	0.04	36	7.61e-13	0.95	40	4.5e-15	0
pilot4	0.03	31	1.80e-13	0.3	31	1.5e-13	0
pilot87	0.80	99	1.21e-08	27.58	42	2.8e-15	0
pilotnov	0.06	20	1.73e-13	1.86	24	1.3e-13	0
recipe	0.01	11	1.32e-13	0.01	11	6.1e-15	0
sc105	0.01	11	4.42e-16	0.01	10	6.0e-16	0
sc205	0.01	11	2.26e-13	0.02	10	7.2e-13	0
sc50a	0.01	10	3.34e-15	0.01	10	5.3e-16	0
sc50b	0.01	8	1.35e-15	0.01	8	6.1e-16	0
scagr25	0.01	17	7.46e-15	0.04	16	3.0e-15	0
scagr7	0.01	13	2.50e-13	0.01	13	7.5e-16	0
scfxm1	0.01	18	1.79e-13	0.06	18	2.0e-15	8
scfxm2	0.02	21	4.24e-14	0.13	20	3.3e-15	16
scfxm3	0.03	21	1.21e-14	0.19	20	3.5e-15	24
scorpion	0.01	15	1.99e-13	NA	NA	NA	132*
scrs8	0.02	26	7.17e-13	0.1	25	6.2e-13	0
scsd1	0.01	10	6.40e-13	0.12	11	3.3e-14	0
scsd6	0.02	15	7.31e-15	0.42	15	6.1e-15	0
scsd8	0.03	12	1.07e-14	2.64	13	2.2e-15	0
sctap1	0.01	17	5.67e-13	0.05	18	2.6e-14	0

Table 4.9: NETLIB set with LIPSOL and Stable Direct method continued

problems	LIPSOL			Stable Direct			
Name	D_time	its	error	D_time	its	error	no-tri-size of B
sctap2	0.03	19	7.33e-13	0.27	16	1.9e-15	0
sctap3	0.04	18	1.46e-13	0.36	21	1.9e-15	0
seba	0.10	23	8.39e-07	0.1	17	7.4e-15	0
share1b	0.01	21	1.92e-13	0.03	24	5.5e-15	66
share2b	0.01	14	5.69e-15	0.01	12	1.2e-14	0
shell	0.02	20	1.61e-15	0.04	12	1.2e-15	494*
ship04l	0.02	13	1.88e-13	0.24	13	1.9e-15	0
ship04s	0.02	14	2.76e-13	0.14	13	1.7e-15	0
ship08l	0.04	16	3.34e-15	0.49	16	2.4e-15	0
ship08s	0.02	14	2.47e-13	0.2	15	2.0e-15	0
ship12l	0.05	17	9.98e-13	0.62	17	1.0e-14	0
ship12s	0.02	19	3.94e-15	0.21	16	3.7e-15	0
sierra	0.06	17	1.50e-13	0.17	12	5.5e-15	515*
stair	0.02	15	2.93e-13	0.1	14	4.8e-13	0
standata	0.02	17	1.62e-14	0.13	17	4.5e-15	0
standgub	0.02	17	5.15e-13	0.06	17	4.0e-15	1*
standmps	0.02	24	9.87e-14	0.19	23	1.7e-14	0
stocfor1	0.01	16	6.84e-13	0.01	19	3.9e-14	0
stocfor2	0.05	22	1.19e-13	0.32	22	1.8e-13	0
tuff	0.02	23	2.83e-16	0.13	20	1.4e-13	0
vtp.base	0.01	23	5.76e-13	0.03	27	3.5e-13	0
wood1p	0.15	21	4.37e-13	0.76	13	6.4e-14	241*
woodw	0.11	30	6.13e-13	41.59	30	9.6e-14	0

Table 4.10: NETLIB set with LIPSOL and Stable Direct method continued

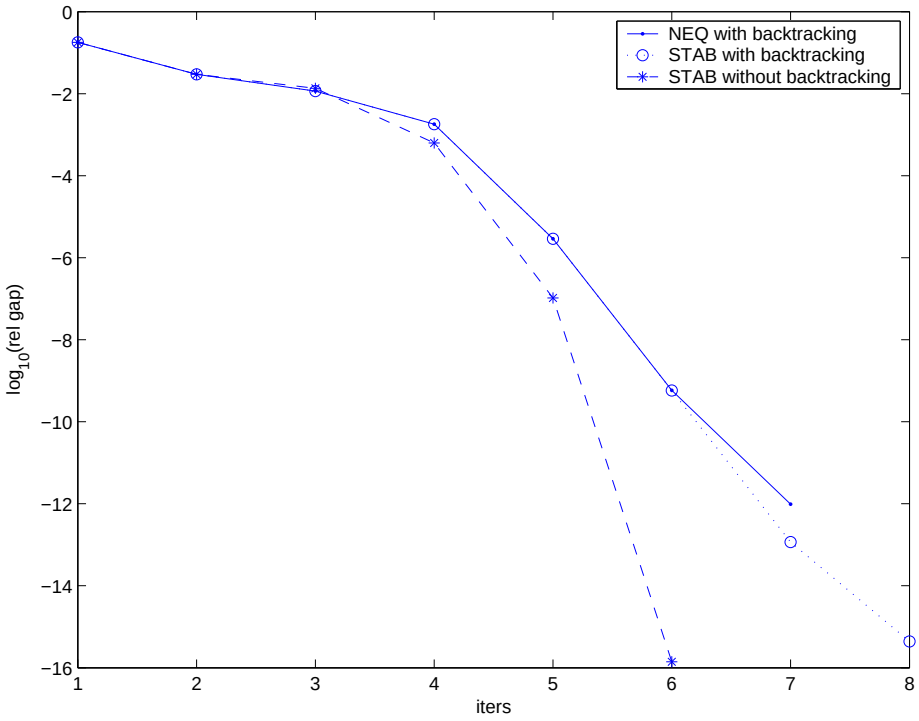

Figure 4.3: Iterations for Different Backtracking Strategies. The data is from row 2 in Table 4.1.

1. under primal and dual nondegeneracy, the resulting linear system for the search direction does not necessarily get ill-conditioned as we approach the optimum;
2. when the linear system is well-conditioned, one may successfully apply: preconditioned iterative methods, a dynamic change to affine scaling without backtracking, dynamic purification, and no backtracking from the boundary (taking the complete step to the boundary is advantageous);
3. high accuracy solutions are obtained for both nondegenerate and degenerate problems; though for ill-conditioned problems this can be be at the expense of (sometimes significantly) larger computational time;
4. exact primal-dual feasibility is maintained throughout the iterations, if we start feasible.

Since our reduced linear system is larger than the usual normal equations approach, $\boldsymbol{NEQ}$, our method is not competitive for the highly ill-conditioned NETLIB test set, with respect to CPU time, though we can obtain higher accuracy solutions. We think that improvements in our preliminary methods for finding B and for the preconditioning in LSQR will result in improved speed and accuracy.

In summary, we believe that our stable approach for interior-point methods for ***LP***s provides: a first step towards greater reliability; and a means for applying iterative methods for finding the search direction. Our method has advantages in comparison with the ***NEQ*** approach when the nondegeneracy assumptions are satisfied or when higher accuracy solutions are needed. Our numerical tests show that we can take direct advantage of sparsity for large sparse well-conditioned problems.

Acknowledgments
The authors are indebted to Michael Saunders (Department of Management Science and Engineering, Stanford University) for providing the *LSQR* MATLAB code for the PCG-like method. The authors would also like to thank Tamas Terlaky (Department of Computing and Software, McMaster University) for many helpful conversations. In addition, we thank an anonymous associate editor and three referees for helping us make numerous significant improvements to the paper.

References

[1] E.D. ANDERSEN and Y. YE. Combining interior-point and pivoting algorithms for linear programming. *Management Science*, 42:1719–1731, 1996.

[2] K.M. ANSTREICHER. Linear programming in $O((n^3/\ln n)L)$ operations. *SIAM J. Optim.*, 9(4):803–812 (electronic), 1999. Dedicated to John E. Dennis, Jr., on his 60th birthday.

[3] A. BEN-TAL and A. NEMIROVSKI. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Program.*, 88(3, Ser. A):411–424, 2000.

[4] A. BEN-TAL and A.S. NEMIROVSKI. Robust convex optimization. *Math. Oper. Res.*, 23(4):769–805, 1998.

[5] A. BEN-TAL and A.S. NEMIROVSKI. Robust solutions of uncertain linear programs. *Oper. Res. Lett.*, 25(1):1–13, 1999.

[6] M. BENZI, C.D. MEYER, and M. TŮMA. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17(5):1135–1149, 1996.

[7] M. BENZI and M. TŮMA. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19(3):968–994, 1998.

[8] L. BERGAMASCHI, J. GONDZIO, and G. ZILLI. Preconditioning indefinite systems in interior point methods for optimization. *Comput. Optim. Appl.*, 28(2):149–171, 2004.

[9] Å. BJÖRCK. Methods for sparse least squares problems. In J.R. Bunch and D. J. Rose, editors, *Sparse Matrix Computations*, pages 177–199. Academic Press, New York, 1976.

[10] J. BURKE. On the identification of active constraints. II. The nonconvex case. *SIAM J. Numer. Anal.*, 27(4):1081–1103, 1990.

[11] J.V. BURKE and J.J. MORÉ. On the identification of active constraints. *SIAM J. Numer. Anal.*, 25(5):1197–1211, 1988.

[12] J.V. BURKE and J.J. MORÉ. Exposing constraints. *SIAM J. Optim.*, 4(3):573–595, 1994.

[13] J-S CHAI and K-C TOH. Preconditioning and iterative solution of symmetric indefinite linear systems arising from interior point methods for linear programming. *Comput. Optim. Appl.*, 36(2-3):221–247, 2007.

[14] R. DE LEONE and O.L. MANGASARIAN. Serial and parallel solution of large scale linear programs by augmented Lagrangian successive overrelaxation. In *Optimization, parallel processing and applications (Oberwolfach, 1987 and Karlsruhe, 1987)*, volume 304 of *Lecture Notes in Econom. and Math. Systems*, pages 103–124. Springer, Berlin, 1988.

[15] J.E. DENNIS Jr. and R.B. SCHNABEL. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. Corrected reprint of the 1983 original.

[16] J.E. DENNIS Jr. and H. WOLKOWICZ. Sizing and least-change secant methods. *SIAM J. Numer. Anal.*, 30(5):1291–1314, 1993.

[17] J. DOMINGUEZ and M.D. GONZALEZ-LIMA. A primal-dual interior-point algorithm for quadratic programming. *Numerical Algorithms*, 105:1–30, 2006.

[18] A.S. EL-BAKRY, R.A. TAPIA, and Y. ZHANG. A study of indicators for identifying zero variables in interior-point methods. *SIAM Rev.*, 36(1):45–72, 1994.

[19] A.V. FIACCO and G.P. McCORMICK. *Nonlinear programming sequential unconstrained minimization techniques*. Classics in Applied Mathematics. SIAM, Philadelphia, PA, USA, 1990.

[20] R.W. FREUND, M.H. GUTKNECHT, and N.M. NACHTIGAL. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM Journal on Scientific Computing*, 14:137–158, 1993.

[21] R.W. FREUND and F. JARRE. A QMR-based interior-point algorithm for solving linear programs. *Mathematical Programming, Series B*, 76:183–210, 1996.

[22] G. H. GOLUB and V. PEREYRA. The differentiation of pseudoinverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.*, 10:413–432, 1973.

[23] N.I.M. GOULD, D. ORBAN, A. SARTENAER, and Ph.L. TOINT. Componentwise fast convergence in the solution of full-rank systems of nonlinear equations. Tr/pa/00/56, CERFACS, Toulouse Cedex 1, France, 2001.

[24] A. GREENBAUM. *Iterative methods for solving linear systems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[25] O. GÜLER, D. DEN HERTOG, C. ROOS, T. TERLAKY, and T. TSUCHIYA. Degeneracy in interior point methods for linear programming: a survey. *Ann. Oper. Res.*, 46/47(1-4):107–138, 1993. Degeneracy in optimization problems.

[26] W.W. HAGER. The dual active set algorithm and the iterative solution of linear programs. In *Novel approaches to hard discrete optimization (Waterloo, ON, 2001)*, volume 37 of *Fields Inst. Commun.*, pages 97–109. Amer. Math. Soc., Providence, RI, 2003.

[27] J.J. JÚDICE, J. PATRICIO, L.F. PORTUGAL, M.G.C. RESENDE, and G. VEIGA. A study of preconditioners for network interior point methods. *Comput. Optim. Appl.*, 24(1):5–35, 2003.

[28] L.V. KANTOROVICH. Functional analysis and applied mathematics. *Uspekhi Mat. Nauk.*, 3:89–185, 1948. Transl. by C. Benster as N.B.S. Rept. 1509, Washington D.C., 1952.

[29] C. KEIL and C. JANSSON. Computational experience with rigorous error bounds for the Netlib linear programming library. *Reliab. Comput.*, 12(4):303–321, 2006.

[30] S. LU and J.L. BARLOW. Multifrontal computation with the orthogonal factors of sparse matrices. *SIAM J. Matrix Anal. Appl.*, 17(3):658–679, 1996.

[31] O.L. MANGASARIAN. Iterative solution of linear programs. *SIAM J. Numer. Anal.*, 18(4):606–614, 1981.

[32] P. MATSTOMS. *The Multifrontal Solution of Sparse Linear Least Squares Problems*. Licentiat thesis, Department of Mathematics, Linköping University, Sweden, 1991.

[33] P. MATSTOMS. Sparse QR factorization in MATLAB. *ACM Trans. Math. Software*, 20:136–159, 1994.

[34] S. MEHROTRA. Implementations of affine scaling methods: approximate solutions of systems of linear equations using preconditioned conjugate gradient methods. *ORSA J. Comput.*, 4(2):103–118, 1992.

[35] S. MEHROTRA and J.-S. WANG. Conjugate gradient based implementation of interior point methods for network flow problems. In *Linear and nonlinear conjugate gradient-related methods (Seattle, WA, 1995)*, pages 124–142. SIAM, Philadelphia, PA, 1996.

[36] S. MEHROTRA and Y. YE. Finding an interior point in the optimal face of linear programs. *Math. Programming*, 62(3, Ser. A):497–515, 1993.

[37] A.R.L. OLIVEIRA and D.C. SORENSEN. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. *Linear Algebra Appl.*, 394:1–24, 2005.

[38] F. ORDÓÑEZ and R.M. FREUND. Computational experience and the explanatory value of condition measures for linear optimization. *SIAM J. Optim.*, 14(2):307–333 (electronic), 2003.

[39] C.C. PAIGE and M.A. SAUNDERS. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71, 1982.

[40] J. PANG. Error bounds in mathematical programming. *Math. Programming*, 79(1-3, Ser. B):299–332, 1997. Lectures on mathematical programming (ismp97) (Lausanne, 1997).

[41] S. PEREZ-GARCIA. Alternative iterative primal-dual interior-point algorithms for linear programming. Master's thesis, Simon Bolivar University, Center for Statistics and Mathematical Software (CESMa), Venezuela, 2003.

[42] S. PEREZ-GARCIA and M. GONZALEZ-LIMA. On a non-inverse approach for solving the linear systems arising in primal-dual interior point methods for linear programming. Technical Report 2004-01, Simon Bolivar University, Center for Statistical and Mathematical Software, Caracas, Venezuela, 2004.

[43] A.N. TIKHONOV and V.Y. ARSENIN. *Solutions of Ill-Posed Problems*. V.H. Winston & Sons, John Wiley & Sons, Washington D.C., 1977. Translation editor Fritz John.

[44] A. VAN der SLUIS. Condition numbers and equilibration of matrices. *Numer. Math.*, 14:14–23, 1969/1970.

[45] R.J. VANDERBEI. *Linear Programming: Foundations and Extensions*. Kluwer Acad. Publ., Dordrecht, 1998.

[46] R.J. VANDERBEI. LOQO: an interior point code for quadratic programming. *Optim. Methods Softw.*, 11/12(1-4):451–484, 1999. Interior point methods.

[47] H. WEI. *Robust Solutions for Large Sparse Linear and Semidefinite Programming*. PhD thesis, University of Waterloo, 2006.

[48] H. WOLKOWICZ. Solving semidefinite programs using preconditioned conjugate gradients. *Optim. Methods Softw.*, 19(6):653–672, 2004.

[49] M.H. WRIGHT. Ill-conditioning and computational error in interior methods for nonlinear programming. *SIAM J. Optim.*, 9(1):84–111 (electronic), 1999.

[50] S. WRIGHT. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa, 1996.

[51] S. WRIGHT. Modifying SQP for degenerate problems. Technical report, Argonne National Laboratory, 1997.

[52] S.J. WRIGHT. Stability of linear equations solvers in interior-point methods. *SIAM J. Matrix Anal. Appl.*, 16(4):1287–1307, 1995.

[53] S.J. WRIGHT. Stability of augmented system factorizations in interior-point methods. *SIAM J. Matrix Anal. Appl.*, 18(1):191–222, 1997.

[54] Y. ZHANG. User's guide to LIPSOL: linear-programming interior point solvers V0.4. *Optim. Methods Softw.*, 11/12(1-4):385–396, 1999. Interior point methods.