SIAM J. OPTIMIZATION
Vol. 2, No. 4, pp. 575-601, November 1992

ON THE IMPLEMENTATION OF A PRIMAL-DUAL INTERIOR POINT METHOD*

SANJAY MEHROTRA†

Abstract. This paper gives an approach to implementing a second-order primal-dual interior point method. It uses a Taylor polynomial of second order to approximate a primal-dual trajectory. The computations for the second derivative are combined with the computations for the centering direction. Computations in this approach do not require that primal and dual solutions be feasible. Expressions are given to compute all the higher-order derivatives of the trajectory of interest. The implementation ensures that a suitable potential function is reduced by a constant amount at each iteration.

There are several salient features of this approach. An adaptive heuristic for estimating the centering parameter is given. The approach used to compute the step length is also adaptive. A new practical approach to compute the starting point is given. This approach treats primal and dual problems symmetrically.

Computational results on a subset of problems available from *netlib* are given. On mutually tested problems the results show that the proposed method requires approximately 40 percent fewer iterations than the implementation proposed in Lustig, Marsten, and Shanno [*Tech. Rep.* TR J-89-11, Georgia Inst. of Technology, Atlanta, 1989]. It requires approximately 50 percent fewer iterations than the dual affine scaling method in Adler, Karmarkar, Resende, and Veiga [*Math. Programming*, 44 (1989), pp. 297-336], and 35 percent fewer iterations than the second-order dual affine scaling method in the same paper. The new approach for estimating the centering parameter and finding the step length and the starting point have contributed to the reduction in the number of iterations. However, the contribution due to the use of second derivative is most significant.

On the tested problems, on the average the implementation shown was found to be approximately two times faster than OB1 (version 02/90) described in Lustig, Marsten, and Shanno and 2.5 times faster than MINOS 5.3 described in Murtagh and Saunders [*Tech. Rep.* SOL 83-20, Dept. of Operations Research, Stanford Univ., Stanford, CA, 1983].

Key words. linear programming, interior point methods, primal-dual methods, power series methods, predictor-corrector methods

AMS(MOS) subject classifications. 90C05, 90C06, 90C20, 49M15, 49M35

1. Introduction.

This paper considers interior point algorithms for simultaneously solving the primal linear program:

$$(P) \qquad \begin{aligned} \text{minimize} \quad & c^T x \\ \text{s.t.} \quad & Ax = b, \\ & x \geqq 0, \end{aligned}$$

and its dual

$$(D) \qquad \begin{aligned} \text{maximize} \quad & b^T \pi \\ \text{s.t.} \quad & A^T \pi + s = c, \\ & s \geqq 0, \end{aligned}$$

where $c, x, s \in \Re^n$, $\pi, b \in \Re^m$, and $A \in \Re^{m \times n}$. It is assumed that A has full row rank. This can be ensured by removing the linearly dependent rows in the beginning. The

* Received by the editors June 18, 1990; accepted for publication (in revised form) August 30, 1991. This paper was presented at the Second Asilomar Workshop on Progress in Mathematical Programming, Monterey, CA, February 5-7, 1990. This research was supported in part by National Science Foundation research initiation grant CCR-8810107 and by a grant from the GTE Laboratories.

† Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208 (mehrotra@iems.nwu.edu).

primal-dual methods, which use solutions of both (P) and (D) in the scaling matrix, are of primary interest.

The primal-dual algorithms have their roots in Megiddo [21]. These were further developed and analyzed by Kojima, Mizuno, and Yoshise [15] and Monteiro and Adler [27]. They showed that the central trajectory can be followed to the optimal solution in $O(\sqrt{n}L)$ iterations by taking "short steps." Kojima, Mizuno, and Yoshise [16] showed that the primal-dual potential function [31], which is a variant of Karmarkar's potential function [13], can also be reduced by a constant amount at each iteration and therefore, they developed a primal-dual large step potential reduction algorithm.

McShane, Monma, and Shanno [20] were the first to develop an implementation of this method. They found it to be a viable alternative to the then popular dual affine scaling method [1], [26] for solving large sparse problems. They also found that this method typically takes fewer iterations than the dual affine scaling method. However, it was found to be only competitive with the dual affine scaling method because of the additional computations that their implementation had to perform. This implementation created some artificial problems (by adding artificial variables/constraints to the original problem) and maintained primal and dual feasible solutions of these problems. Further developments on this implementation were reported in Choi, Monma, and Shanno [4].

Lustig, Marsten, and Shanno [18] implemented a variant of the primal-dual method which is based on an earlier work of Lustig [17]. This method is developed by considering the Newton direction on the optimality conditions for the logarithmic barrier problem. An important feature of their approach is that it did not explicitly require that feasible solutions for the primal or the dual problem be available. They showed that the resulting direction is a particular combination of primal-dual affine scaling direction, feasibility direction, and centering direction. In support of their method they reported success in solving all the problems in the *netlib* [7] test set.

This paper builds on the work of Lustig, Marsten, and Shanno [18]. In doing so it makes use of the work of Monteiro, Adler, and Resende [28] and Karmarkar, Lagarias, Slutsman, and Wang [14]. The discussion in this paper assumes that direct methods are preferred over iterative methods for solving linear equations arising at each iteration of the algorithm. In our view, the following accomplishments are reported in this paper:

- It gives an algorithm and describes its implementation by using first and second derivatives of the primal-dual affine scaling trajectory Taylor polynomial and by effectively combining the second derivative with a centering direction.
- A comparison with the results reported in the literature (on mutually tested problems) shows that the method developed in this paper takes approximately 50 percent fewer iterations than the dual affine scaling method as implemented by Adler, Karmarkar, Resende, and Veiga [1], 40 percent fewer iterations than the primal-dual method implemented in Lustig, Marsten, and Shanno [18], and 55 percent fewer iterations than the logarithmic barrier function method implemented in Gill, Murray, and Saunders [8]. It requires 35 percent fewer iterations than the second-order dual affine scaling method implemented in Adler, Karmarkar, Resende, and Veiga [1] and 20 percent fewer iterations than the "optimal three-dimensional method" implemented by Domich, Boggs, Donaldson, and Witzgall [5].
- An efficient preliminary implementation of the proposed approach was developed. On average, it was found to be two times faster than OB1 (version 02/1990) [18]. On average, it was also found to be 2.5 times faster than MINOS 5.3.

- While developing our implementation we ensure that a suitable potential function is reduced by a constant amount. This is accomplished by taking a two tier approach. Most of the work is performed at the first level, which uses extensive heuristic arguments. The second level ensures the robustness of the implementation.
- It gives expressions for computing first and higher derivatives of a primal-dual trajectory. This trajectory starts from any positive point and goes to the optimum.
- It gives an adaptive approach to computing the centering parameter.
- It gives a modified heuristic for computing step length at each iteration. The approach allows us to adaptively take steps much closer to the boundary.
- It gives an approach to generating primal and dual starting points, which treat these problems symmetrically.

We find it convenient to outline the proposed approach first. This is done in the next section. The organization of this paper is given in that section. The following notation and terminology is used throughout this paper.

Notation and terminology. x^k, π^k, and s^k represent the estimate of solutions of (P) and (D) at the beginning of iteration k. X^k and S^k are used to represent diagonal matrices whose elements are $x_1^k, x_2^k, \ldots, x_n^k$ and $s_1^k, s_2^k, \ldots, s_n^k$, respectively. $\xi_x^k = Ax^k - b$, $\xi_s^k = A^T\pi^k + s^k - c$, ξ_x, and ξ_s are referred to as *error vectors*. D^2 represents matrix $(S^k)^{-1}X^k$. e is used to represent a vector of all ones. e_i represents column i of an identity matrix. $\| \ \|$ is used to represent the Euclidean norm of a vector.

The term *search direction in primal space* is used for a direction p_x, which is constructed from a combination of directions (p_x1, p_x2). The term *primal blocking variable* is used for a variable that will first become negative when moving in a direction. The term *step factor* represents the fraction of step length that makes the blocking variable zero. Similar terminology is used for directions in the dual space.

Central trajectory is the set of feasible points in (P) and (D) satisfying $x_i(\mu)s_i(\mu) = \mu$, for $i = 1, \ldots, n$. In all references to central trajectory we assume that $x(\mu)$, $s(\mu)$ exists for all μ.

2. Implementation of an interior point method. This section outlines the approach we take to implement an interior point method. We do this to provide a complete picture of this paper and to fix certain additional notations used throughout. Various steps in the development of this implementation are discussed in more detail in §§ 3–7. The procedure is outlined in Exhibit 2.1 and it is called AIPM (an interior point method). We now discuss the procedure.

Procedure AIPM
Input: Let $x^0 > 0$ and $s^0 > 0$, π^0 be the given starting points.
For $k = 0, 1, \ldots$ until a stopping criterion is satisfied **do:**
Step 0

$$\xi_s^k := A^T\pi^k + s^k - c,$$
$$\xi_x^k := Ax^k - b,$$
$$D^2 := S^{k^{-1}}X^k.$$

Step 1
c find the first derivative of primal-dual affine scaling trajectory.

$$p_\pi 1 := -(AD^2A^T)^{-1}(b - AD^2\xi_s^k),$$
$$p_s 1 := \xi_s^k - A^T p_\pi 1,$$
$$p_x 1 := x^k - D^2 p_s 1.$$

EXHIBIT 2.1. *A pseudo-code for implementing a second-order primal-dual method.*

Step 2

c compute centering parameter μ^k.

$$\text{CALL CENPAR}\,(x^k, p_x1, s^k, p_s1, \mu^k).$$

c compute the second derivative of primal-dual trajectory with the centering direction.

Step 3

$$v_i := -2 * ((p_x1)_i * (p_s1)_i - \mu^k)/s_i^k \text{ for } i = 1, 2 \ldots n,$$
$$p_\pi 2 := (AD^2A^T)^{-1}Av,$$
$$p_s 2 := A^T p_\pi 2,$$
$$p_x := v - D^2 p_s 2.$$

c construct a Taylor polynomial and find maximum steps $(\varepsilon_x, \varepsilon_s)$ using this polynomial.

Step 4

$$\text{CALL SFSOP}\,(x^k, p_x1, p_x2, \varepsilon_x, s^k, p_s1, p_s2, \varepsilon_s).$$

c construct a search direction.

Step 5

$$p_s := \varepsilon_s * p_s1 - .5 * \varepsilon_s^2 * p_s2,$$
$$p_\pi := \varepsilon_s * p_\pi 1 - .5 * \varepsilon_s^2 * p_s2,$$
$$p_x := \varepsilon_x * p_x1 - .5 * \varepsilon_x^2 * p_x2.$$

c compute step factors (f_x, f_s).

Step 6

$$\text{CALL GTSF}\,(x^k, p_x, s^k, p_s, f_x, f_s).$$

c generate trial points.

Step 7

$$\hat{x} := x^k - f_x * p_x,$$
$$\hat{s} := s^k - f_s * p_s,$$
$$\hat{\pi} := \pi^k - f_s * p_\pi.$$

c test if the trial point is acceptable.

Step 8

If an appropriate potential function is reduced, then

$$x^{k+1} := \hat{x},$$
$$s^{k+1} := \hat{y},$$
$$\pi^{k+1} := \hat{\pi},$$

else

perform a line search/if necessary compute
additional vectors and ensure reduction in the potential function.

endif

EXHIBIT 2.1 (*continued*).

The approach used to generate a starting point is discussed in § 7.

Given x^k, π^k, and s^k, Step 0 computes error vectors ξ_x^k and ξ_s^k representing the amount by which primal and dual constraints are violated. D^2 has the primal-dual scaling matrix.

Step 1 computes direction $p_x 1$ in primal and $p_\pi 1$, $p_s 1$ in dual spaces. These directions are tangent to a primal-dual trajectory. This trajectory is discussed in § 4. Expressions to compute all derivatives of this trajectory at a point are also developed in § 4.

The primal and dual directions computed at Step 1 are used in procedure CENPAR to estimate the centering parameter μ^k. Our approach to estimating the centering parameter is given in Exhibit 5.1. This approach is discussed in § 5.

Step 3 computes the second derivative of the primal-dual trajectory and the centering direction. These directions could be computed separately. However, in the current implementation we prefer to combine their computation in order to save a forward and a back solve. We use the tangent direction and the direction in Step 3 to construct a Taylor polynomial and to find a maximum step to the boundary (in primal and dual spaces separately) using this polynomial. This is done in Procedure SFSOP given in Exhibit 4.1. The computations performed in this procedure are also discussed in § 4.

In Step 5 we use the maximum step in a Taylor polynomial to generate search directions p_x, p_π, and p_s. In Procedure GTSF (Exhibit 6.1) we compute a fraction (f_x, f_s) of the total step to the boundary in the search direction. This is discussed further in § 6. Using the search directions and the step factors, trial point $\hat{x}$, $\hat{\pi}$, $\hat{s}$ is generated in primal and dual spaces.

Step 8 ensures the robustness of the overall procedure. It is loosely defined here. It depends on the choice of the function used to measure the progress of the algorithm and the best possible theoretical results that could be proved for this function. The potential function we used to ensure the progress is developed in the next section (§ 3), and our motivations for using it are discussed there.

If the potential function is not reduced by the desired amount at the trial points, we may perform a line search and, if necessary, compute additional directions to ensure a reduction in this function. This actually happened for the potential function we discuss in the next section. If this happens, we generate an additional three trial points by using $\varepsilon_x := \varepsilon_s := \min(\varepsilon_x, \varepsilon_s)$; $\varepsilon_x := \varepsilon_x$, $\varepsilon_s := 0$; and $\varepsilon_x := 0$, $\varepsilon_s := \varepsilon_s$ in Step 5 to compute p_x, p_π, p_s. The potential function was always reduced by the desired amount at one of the new trial points. Therefore, on the tested problems, additional vectors were never computed and explicit line searches were never performed.

3. A potential function. In our view the interior point methods generate one or more interesting search directions at each iteration and effectively combine these directions to ensure that sufficient progress in a suitable convergence function is made. Various proposed methods differ in the directions they compute, in how they combine these directions (implicitly or explicitly), and in the convergence function they use to measure the progress [12]. Unfortunately, to our knowledge, the current theoretical understanding of these methods has not reached a point where a clear superiority of one method is established. Hence, practical implementations [1], [5], [18], [19], [20], [22], [26] rely on heuristic arguments and empirical evidence obtained from performing experiments on a set of real problems.

Use of a suitable potential function is frequently ignored while developing fast implementations. In our experience, an important reason, among others, is that the

cost of performing line searches in one- or higher-dimensional subspaces is significant on sparse problems, and it is frequently not justified by the return.

In our opinion use of heuristic arguments is justified, but not at the cost of the robustness of the solution procedures. Hence, even though in this paper several different heuristics are proposed and their use justified solely on the basis of empirical evidence, in the actual implementation we recommend the use of a potential function.

We now develop the potential function that was used to measure progress in our implementation. We find it instructive to go through some construction to motivate this function. Some steps used in this construction appeared in Karmarkar [13] and others in Goldfarb and Mehrotra [9] and Todd and Ye [31].

Let $x^0 > 0$, π^0, $s^0 > 0$, be any given point. Let $\xi_x^0 = Ax^0 - b$ and $\xi_s^0 = A^T\pi^0 + s^0 - c$. In order to solve (P) and (D), it is enough to find an optimal solution of

$$\begin{aligned} &\text{minimize} && \lambda \\ &\text{s.t.} && Ax - \lambda\xi_x^0 = b, \\ &&& A^T\pi + s - \lambda\xi_s^0 = c, \\ &&& c^Tx - b^T\pi + \lambda(b^T\pi^0 - c^Tx^0) = 0, \\ &&& x_i, s_i, \quad \lambda \geqq 0, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{3.1}$$

$(x^0, \pi^0, s^0, 1)$ is a feasible interior solution of (3.1). Let Z be a matrix whose columns are the basis for the null space of A. Multiplying the second set of equations in (3.1) with $[A^T : Z]^T$ and solving for the free variables π results in

$$\pi = (AA^T)^{-1}(Ac - As + \lambda A\xi_s^0); \tag{3.2}$$

therefore, solving (3.1) is the same as:

$$\begin{aligned} &\text{minimize} && \lambda \\ &\text{s.t.} && Ax - \lambda\xi_x^0 = b, \\ &&& Z^Ts - \lambda Z^T\xi_s^0 = Z^Tc, \\ &&& c^Tx + b^T(AA^T)^{-1}As + \lambda\xi_a = b^T(AA^T)^{-1}Ac, \\ &&& x_i, s_i, \lambda \geqq 0, \end{aligned} \tag{$PD0$}$$

where $\xi_a = (b^T\pi^0 - c^Tx^0 - b^T(AA^T)^{-1}A\xi_s^0)$. Consider the potential function

$$F(x, s, \lambda) = \rho \ln \lambda - \sum_{i=1}^{n} \ln x_i s_i \tag{3.3}$$

for $\rho = 2n + \sqrt{2n+1}$. In the Appendix we show that $F(x, s, \lambda)$ can be reduced by a constant amount (.25) at any feasible solution of $(PD0)$.

Let $\xi^k = ((b - Ax^k)^T, (c - s^k)^TZ, b^T(AA^T)^{-1}A(c - s^k) - c^Tx^k)^T$. Note that ξ^k is λ^k times the last column in $(PD0)$. If x^k, π^k, s^k, λ^k and x^{k+1}, π^{k+1}, s^{k+1}, λ^{k+1} are feasible solutions of $(PD0)$, then

$$\begin{aligned} F(x^{k+1}, s^{k+1}, \lambda^{k+1}) - F(x^k, s^k, \lambda^k) &= \rho \ln \frac{\lambda^{k+1}}{\lambda^k} - \sum_{i=1}^{n} \ln \frac{x_i^{k+1}s_i^{k+1}}{x_i^k s_i^k} \\ &= \rho \ln \frac{\|Q\xi^{k+1}\|}{\|Q\xi^k\|} - \sum_{i=1}^{n} \ln \frac{x_i^{k+1}s_i^{k+1}}{x_i^k s_i^k} \end{aligned}$$

for any nonsingular matrix $Q \in \Re^{(n+1)\times(n+1)}$. An important consequence of this observation is that it ensures that the potential function

$$E(x, s, \xi, Q) = \rho \ln \|Q\xi\| - \sum_{i=1}^{n} \ln x_i s_i \tag{3.4}$$

can be reduced by a constant amount at each iteration. We use the following potential function

$$E(x, s, \xi) = \rho \ln \|(\kappa_x \xi_x^T, \kappa_s \xi_s^T, \xi_a)^T\| - \sum_{i=1}^{n} \ln x_i s_i, \tag{3.5}$$

where κ_x and κ_s are some prespecified constants. The potential function (3.5) is used for the following reasons: (i) We think that a potential function of the form (3.5) is superior for developing implementations because it allows for numerical errors [10]. (ii) It is easily computable without having to know Z. (iii) It allows us to separately update primal and dual solutions and the corresponding error vectors. (iv) There is no unknown that has to be determined during the algorithm. (v) Finally, it is possible to compute directions which ensure that (3.3), and therefore (3.5), is reduced by a constant amount.

The potential function (3.5) is, however, dependent on the scaling of rows (in general, the choice of Q in (3.4)). Because of this, a search direction that may be acceptable while using one scaling matrix may become unacceptable for a different choice. However, in our implementation we use it to our advantage. We think that the construction of directions p_x1, $p_\pi1$, p_s1 and p_x2, $p_\pi2$, p_s2 discussed in the next section is inherently biased towards finding (nearly) feasible solutions first. The values of κ_x and κ_s are chosen so that they emphasize primal and dual feasibility over the feasibility of the last equality constraint in ($PD0$). As a consequence of this, search directions that reduce ξ_x and/or ξ_s significantly, and do not reduce (or possibly increase) the error in the last equality constraint of (3.1) become acceptable.

$\kappa_x = 100 * \max_i \{s_i^0\}$ and $\kappa_s = 100 * \max_i \{x_i^0\}$ were used for all the problems in our implementation. The construction of x^0 and s^0 is described in § 7.

A reduction by constant amount in (3.5) at each iteration ensures convergence to an optimal solution provided that $\sum_{i=1}^{n} \ln x_i s_i$ remain bounded. On the other hand, if (3.5) cannot be reduced by a constant amount at some iteration, then either (P) or (D) or both do not have a feasible solution. We may introduce a constraint providing an upper bound on x and s if we detect (through some tests) that the method is not converging.

4. Derivatives of a primal-dual trajectory. This section provides motivation for using directions p_x1, $p_\pi1$, p_s1, p_x2, $p_\pi2$, p_s2 in Procedure AIPM. It was mentioned that these directions use first and second derivative information of a primal-dual trajectory at a given point. This section defines the trajectory of interest and also shows how to compute all of its derivatives at a given point. While we used the potential function (3.5) to measure the progress, derivatives of the primal-dual trajectory being considered are used because they are easily computed and found to be effective in practice.

The results in Monteiro, Adler, and Resende [28] are used frequently to develop these expressions. Monteiro, Adler, and Resende [28] assume that feasible solutions are available. We do not assume this here. The expressions are given in the context of linear programming problems. Extensions to convex quadratic programming are straightforward.

Assume that $x^k > 0$, π^k, $s^k > 0$ is the current point. Consider the following system of nonlinear equations:

$$\begin{aligned} X(\alpha)s(\alpha) &= \alpha X^k s^k, \\ Ax(\alpha) &= b + \alpha\xi_x^k, \\ A^T\pi(\alpha) + s(\alpha) &= c + \alpha\xi_s^k, \\ x(\alpha) \geqq 0, \quad s(\alpha) &\geqq 0 \end{aligned} \tag{4.1}$$

for $\alpha \in [0, 1]$. Let $w(\alpha) \equiv (x(\alpha), \pi(\alpha), s(\alpha))$ represent the solutions of (4.1) for a given α.

PROPOSITION 4.1. *If the system of equations* (4.1) *has a solution for* $\alpha = 0$, *then it has a solution for all* $\alpha \in [0, 1]$. *Furthermore, the solution is unique for* $\alpha \in (0, 1]$.

Proof. For $\alpha \in (0, 1]$ (4.1) gives the optimality conditions for the weighted logarithmic barrier problems

$$(P_\alpha) \qquad \begin{aligned} \text{minimize} \quad & B(x, \alpha) \equiv (c + \alpha\xi_s^k)^T x - \alpha \sum_{i=1}^{n} x_i^k s_i^k \ln x_i \\ \text{s.t.} \quad & Ax = b + \alpha\xi_x^k, \\ & x > 0, \end{aligned}$$

and

$$(D_\alpha) \qquad \begin{aligned} \text{maximize} \quad & (b + \alpha\xi_x^k)^T \pi + \alpha \sum_{i=1}^{n} x_i^k s_i^k \ln s_i \\ \text{s.t.} \quad & A^T\pi + s = c + \alpha\xi_s^k, \\ & s > 0. \end{aligned}$$

Let $x(0)$, $\pi(0)$, $s(0)$ represent a solution of (4.1) for $\alpha = 0$. For a fixed $\alpha \in [0, 1]$, $\tilde{x}(\alpha) = (1-\alpha)x(0) + \alpha x^k$ is a feasible solution for (P_α) and $\tilde{\pi}(\alpha) = (1-\alpha)\pi(0) + \alpha\pi^k$, $\tilde{s}(\alpha) = (1-\alpha)s(0) + \alpha s^k$ is a feasible solution for (D_α). If the feasible set of (P_α) is bounded, then obviously (P_α) has a solution.

We now consider the case when the feasible set of (P_α) is unbounded. Since $\tilde{\pi}(\alpha)$, $\tilde{s}(\alpha)$ is a feasible solution for (D_α), it can be shown that the set $\{d | Ad = 0,\ d \geqq 0,\ d \neq 0,\ (c + \alpha\xi_s^k)^T d \leqq 0\}$ is empty. Hence, the set $P_t \equiv \{x | Ax = b + \alpha\xi_x^k,\ x > 0,\ (c + \alpha\xi_s^k)^T x = t\}$ is bounded for all values of $t < \infty$. Furthermore, since the feasible region of (P_α) is nonempty and unbounded, P_t is nonempty for all values of $t > t^*$, where t^* is the minimum value of $(c + \alpha\xi_s^k)^T x$ subject to $Ax = b + \alpha\xi_x^k$, $x \geqq 0$. Clearly, $B(x, \alpha)$ is bounded over P_t for all values of t. Now, to complete the proof for the existence of the solution, note that in $B(x, \alpha)$, $(c + \alpha\xi_s^k)^T x$ increases linearly in t, while $\max \sum_{i=1}^{n} (x_i^k s_i^k) \ln x_i$ subject to $\{x | Ax = b + \alpha\xi_x^k,\ x > 0,\ (c + \alpha\xi_s^k)^T x = t\}$ increases only logarithmically in t.

The proof for the uniqueness of solution follows from the strict convexity of $B(x, \alpha)$. □

Let $d^j w(\alpha)/d\alpha^j$ be the jth derivative of $w(\alpha)$. It is now shown that $d^j w(\alpha)/d\alpha^j$ can be computed for all j at $\alpha = 1$. Differentiating (4.1) for the first time gives

$$\begin{aligned} S(\alpha)\frac{dx(\alpha)}{d\alpha} + X(\alpha)\frac{ds(\alpha)}{d\alpha} &= X^k s^k, \\ A\frac{dx(\alpha)}{d\alpha} &= \xi_x^k, \\ A^T\frac{d\pi(\alpha)}{d\alpha} + \frac{ds(\alpha)}{d\alpha} &= \xi_s^k. \end{aligned} \tag{4.2}$$

The solution of (4.2) at $\alpha = 1$ is given by

$$\begin{aligned} \frac{d\pi(1)}{d\alpha} &= -(AD^2A^T)^{-1}(b - AD^2\xi_s^k), \\ \frac{ds(1)}{d\alpha} &= \xi_s^k - A^T\frac{d\pi(1)}{d\alpha}, \\ \frac{dx(1)}{d\alpha} &= x^k - D^2\frac{ds(1)}{d\alpha}. \end{aligned} \tag{4.3}$$

Further differentiating (4.2) gives

$$\begin{aligned} &\sum_{l=0}^{j}\binom{j}{l}\frac{d^l x_i(1)}{d\alpha^l}\frac{d^{(j-l)}s_i(1)}{d\alpha^{(j-l)}} = 0, \quad j \geqq 2, \quad i = 1, \ldots, n, \\ &A\frac{d^j x(1)}{d\alpha^j} = 0, \\ &A^T\frac{d^j\pi(1)}{d\alpha^j} + \frac{d^j s(1)}{d\alpha^j} = 0. \end{aligned} \tag{4.4}$$

From (4.4) it is clear that $d^j w(\alpha)/d\alpha^j$ can be computed recursively. The derivatives can be computed explicitly from

$$\begin{aligned} \frac{d^j\pi(1)}{d\alpha^j} &= -(AX^kS^{k^{-1}}A^T)^{-1}AS^{k^{-1}}u, \\ \frac{d^j s(1)}{d\alpha^j} &= -A^T\frac{d^j\pi(1)}{d\alpha^j}, \\ \frac{d^j x(1)}{d\alpha^j} &= S^{k^{-1}}u + S^{k^{-1}}X^k\frac{d^j s(1)}{d\alpha^j}, \\ u_i &= -j!\sum_{l=1}^{j-1}\left(\frac{d^l x(1)}{d\alpha^l}\right)i\left(\frac{d^{(j-1)}s(1)}{d\alpha^{(j-1)}}\right)i, \quad i = 1, \ldots, n; \quad j \geqq 2. \end{aligned} \tag{4.5}$$

The recursion (4.5) is the same as the recursion given in Monteiro, Adler, and Resende [28] and Karmarkar et al. [14] (see also Megiddo [21] and Bayer and Lagarias [2]). The derivatives resulting from the computations would be the same if $\xi_x^k = 0$ and $\xi_s^k = 0$ is assumed, and in the latter paper if no centering and reparameterization is done.

The point $w(1-\varepsilon)$ for $1 > \varepsilon > 0$ can be approximated by using the rth-order Taylor polynomial

$$w((1-\varepsilon), r) \equiv w(1) + \sum_{j=1}^{r}\frac{(-\varepsilon)^j}{j!}\frac{d^j w(1)}{d\alpha^j}. \tag{4.6}$$

The Taylor polynomial is considered for the following two reasons.

(1) In the special case Monteiro, Adler, and Resende [28] established powerful results (near the central path) for this approximation.

(2) Computational results indicate that near the optimal solution the first- and second-order approximations result in nearly unit steps. Our results also indicate that, asymptotically, $w(\alpha)$ is well represented by the Taylor polynomial of a lower order. In this context Megiddo [21] has argued that for problems with unique optimal solution, if we start close to an optimal solution, the primal-dual paths take us approximately in a straight line to the optimal solution. Most of the tested problems do not satisfy this assumption, however they still show this property.

In addition to other things, practical implementations that compute more than one direction must offset the cost of doing extra work. Adler et al. [1] were the first to show that in the dual affine scaling method the information from a second derivative can be used to significantly reduce the number of iterations. However, on problems in the *netlib* test set they found that reduction in the number of iterations did not always translate into reduction in cpu time on sparse problems. Computational results were also given in Karmarkar et al. [14] on a small set of "representative problems" using methods implemented in the AT&T KORBX system [3].

This paper restricts itself to using the second-order Taylor polynomial. In fact, we compute only two directions. The tangent direction $d^j w(\alpha)/d\alpha$ is computed at Step 1 of Procedure AIPM. Step 3 combines the computation of a second derivative with that of a centering direction. This saves a forward and a back solve. We must compute two directions at each iteration in order to use the adaptive approach for computing the centering parameter (§ 5).

Our strategy of combining the computations for second derivative and the centering direction seem to work in practice for the following reasons. (i) The performance of interior point methods in practice weakly depends on the choice of centering parameter. (ii) If we view the computations in constructing the Taylor polynomial as that of finding a search direction, then in practice it appears that a wide range of ε can be used without adversely affecting the performance of the implementation. To illustrate this, we would like the reader to compare the iteration counts reported in Mehrotra [24], [25] for a predictor-corrector method with those in Table 8.2. The predictor-corrector method results if we take $\varepsilon = 1$ at each iteration.

We find that taking different steps in primal and dual spaces generally results in superior performance. This is similar to the experience of Choi, Monma, and Shanno [4] for their method. We construct different polynomials

$$x(\varepsilon_2, 2) \equiv x^k - \varepsilon_x p_x 1 + \varepsilon_x^2 p_x 2, \tag{4.7}$$

$$s(\varepsilon_s, 2) \equiv s^k - \varepsilon_s p_s 1 + \varepsilon_s^2 p_s 2 \tag{4.8}$$

in primal and dual spaces. The computations for ε_x and ε_s that use (4.7)-(4.8) are described in Procedure SFSOP (step from second-order polynomial) of Exhibit 4.1. A procedure that finds the root of a quadratic equation is used to implement SFSOP.

Procedure SFSOP $(x^k, p_x 1, p_x 2, \varepsilon_x, s^k, p_s 1, p_s 2, \varepsilon_s)$
Find maximum $0 \leqq \varepsilon_x \leqq 1$ such that $x(\varepsilon_x, 2)$ is feasible.
Find maximum $0 \leqq \varepsilon_s \leqq 1$ such that $s(\varepsilon_s, 2)$ is feasible.

EXHIBIT 4.1. *Computations for step size using the Taylor polynomial.*

Before concluding this section, we point out that if there are reasons to believe that at the current iterate it is better to target a solution satisfying $XS = W$ for some positive diagonal matrix W, then expressions for derivatives of a trajectory taking us to such a point can be obtained in a similar manner. The only difference would be to replace "$\alpha X^k s^k$" in (4.1) with "$\alpha X^k s^k + (1-\alpha) We$." In particular, we may use the Heuristic CENPAR to compute the centering parameter (given in the next section) in order to decide a target point on the central path, then go back and find desired derivatives of a trajectory going to this point.

5. Centering. In § 4 expressions were developed to construct a Taylor polynomial at a given point in order to approximate a path going to an optimal solution. Obviously, the performance of the algorithm depends to a great extent on how well a "small-order" Taylor polynomial approximates this path at the current point, and on the domain in which the Taylor polynomial results in good approximations.

The results in Monteiro, Adler, and Resende [28] and the convergence results of large step polynomial time algorithms proved by Freund [6], Gonzaga and Todd [11], and Ye [32] implicitly or explicitly use the properties of the central path. The projected gradient of the potential function used for analysis in these papers encourages centering. On the other hand, it is not clear if the central path (with equal weights) is the best path to follow, particularly since it is affected by the presence of redundant constraints [30]. Furthermore, the points on (or near) the central path are only intermediate to solving the linear programming problem. It is only the limit point on this path that is of interest to us.

In view of this, we make our implementation weakly dependent on centering. The centering direction is obtained by solving the equations

$$\tilde{p}_\pi = \mu^k (AD^2A^T)^{-1} AS^{k^{-1}} e, \quad \tilde{p}_s = A^T \tilde{p}_\pi, \quad \tilde{p}_x = \mu^k S^{k^{-1}} e - D^2 \tilde{p}_s.$$

μ^k is called the centering parameter. It was mentioned in § 4 that the computation for the centering direction is combined with computations for the second derivative to save an extra forward and backward solve.

Heuristic CENPAR given in Exhibit 5.1 was used to compute μ^k. In the description of this heuristic we assume that the direction tangent to the primal-dual affine scaling trajectory has been computed. The heuristic is adaptive. It attempts to generate a value of μ^k, depending on the progress that could be made by moving in the tangent direction.

Heuristic CENPAR $(x^k, p_x1, s^k, p_s1, \mu^k)$

Step 1. Let $\varepsilon_x^1, \varepsilon_s^1$ be computed as follows:

$$\varepsilon_x 1 = \min\left(\frac{x_{lx}^k}{(p_x 1)_{lx}}, 1\right),$$

$$lx = \operatorname{argmin}\left\{\frac{x_i^k}{(p_x 1)_i} \,\middle|\, (p_x 1)_i > 0\right\},$$

$$\varepsilon_s 1 = \min\left(\frac{s_{ls}^k}{(p_s 1)_{ls}}, 1\right),$$

$$ls = \operatorname{argmin}\left\{\frac{s_i}{(p_s 1)_i} \,\middle|\, (p_s 1)_i > 0\right\}.$$

Step 2. Let $mdg = (x - \varepsilon_x 1 p_x 1)^T (s - \varepsilon_s 1 p_s 1)$.

Step 3. Let $\mu^k = \dfrac{x^T s}{n}\left(\dfrac{mdg}{x^T s}\right)^\nu$.

Step 4. Let $ef = \dfrac{(p_x 1)^T D^{-2} (p_x 1) + (p_s 1)^T D^2 (p_s 1)}{x^T s}$.

Step 5. If $(ef > 1.1)\mu^k = \mu^k / \min(\varepsilon_x^1, \varepsilon_s^1)$.

EXHIBIT 5.1. *A heuristic to compute centering parameter.*

The motivation behind various steps in Heuristic CENPAR are now discussed. For the moment assume that $\xi_x = 0$ and $\xi_s = 0$. If this is the case, then $x^T s$ is the current duality gap and mdg is the minimum duality gap that one can achieve by moving in directions $p_x 1$ and $p_s 1$ in primal and dual spaces, respectively. $\varepsilon_x 1$ and $\varepsilon_s 1$ are always taken smaller than one, because at this value the computations for $p_x 1$ and $p_s 1$ ensure that $\xi_x = 0$ and $\xi_s = 0$ if no numerical error is present. Hence, for $\nu = 1$ the choice of μ is such that it targets the point on the central path at which the duality gap is mdg.

The ratio $\text{mdg}/x^T s$ provides us "some indication" of how well the primal-dual affine scaling trajectory is being approximated locally. A value of ratio $\text{mdg}/x^T s$ near 1 means that the local approximations are not good, whereas $\text{mdg}/x^T s$ near zero indicates that the approximations of the trajectory are good.

Table 5.1 gives the number of iterations required to solve the problems for choices of $\nu = 1, 2, 3, 4$. All other parameters were the same as those for results in Table 8.2. The last column of this table gives the number of iterations required to solve the problem if no centering was done. The results in Table 5.1 on the test problems show only a moderate variation in the number of iterations for values of ν between two and four.

The discussion on the computation of the centering parameter thus far assumed that $\xi_x = 0$ and $\xi_s = 0$. If this is not the case, then ef (error factor) is used as an indicator for their contribution to the search direction. If $\xi_x = 0$ and $\xi_s = 0$, then it is easy to see that

$$Dp_s 1 = [DA^T(AD^2A^T)^{-1}AD](X^k S^k)^{1/2} e,$$
$$D^{-1}p_x 1 = [I - DA^T(AD^2A^T)^{-1}AD](X^k S^k)^{1/2} e;$$

hence ef as defined in Step 4 of Exhibit 5.1 is equal to 1. If ef is smaller than 1, it indicates that the presence of ξ_x and ξ_s is probably reducing the norm of the search direction and, therefore, it is expected to allow for larger steps in primal and/or dual spaces. Since ξ_x^k and ξ_s^k reduce linearly in step size when moving in directions $p_x 1$ and $p_s 1$, respectively, larger steps result in greater reduction in the error vectors. Hence, the value of ef smaller than one is not likely to hurt the performance of the implementation.

Now if ef > 1, then empirical results indicate that the presence of ξ_x and/or ξ_s results in a reduction in the step length, which adversely affects the improvement in the duality gap as well as the reduction in the error vectors. Therefore, it might be indicating trouble ahead. If this happens in practice, we seem to quickly get out of the trouble spots by placing more emphasis on centering. In the current implementation this is accomplished in Step 5.

6. Step length. Standard practice [1], [5], [18], [19], [20], [26] has been to move a certain fixed distance (step factor) to the boundary to avoid one-dimensional line searches. The step factor in the case of primal-dual methods has typically been .995 or .9995 [18].

Although the performance of step factor =.995 or .9995 appears satisfactory in practice, in our view, it has a major drawback as a heuristic: it limits the asymptotic rate of convergence of the algorithm. Furthermore, during the earlier phase of the algorithm it is overly aggressive. A modified approach to computing step factor is given in Exhibit 6.1. It adaptively allows for larger (and smaller) step factors. In an extreme case it may allow a full step to the boundary and generate a point with zero duality gap.

TABLE 5.1
Performance of implementation for different choices of ν. +Stopped after 100 iterations with one digit of accuracy in objective function.

	ν				
Problem	1	2	3	4	nc
afiro	9	8	7	7	8
adlittle	14	11	10	10	11
scagr7	17	14	13	13	15
stochfor1	16	16	16	16	21
sc205	14	12	11	11	11
share2b	14	12	12	13	14
share1b	23	22	22	20	25
scorpion	13	12	12	12	12
scagr25	19	17	16	17	18
sctap1	17	15	15	15	17
brandy	21	20	20	19	19
scsd1	9	8	8	8	8
israel	30	25	24	24	26
bandm	20	17	17	19	17
scfxm1	21	19	18	18	19
e226	25	21	20	20	21
agg	27	22	25	27	56
scrs8	24	21	21	21	22
beaconfd	11	8	7	7	7
scsd6	13	10	10	10	10
ship04s	15	12	13	13	15
agg2	23	21	24	23	26
agg3	22	19	21	21	23
scfxm2	22	18	19	19	20
ship041	14	12	12	12	12
fffff800	36	38	38	38	100^+
ship08s	16	13	13	13	13
sctap2	15	13	12	12	13
scfxm3	25	20	20	20	19
ship12s	18	16	16	17	19
scsd8	12	10	9	9	9
czprob	39	33	35	35	38
ship081	18	14	14	14	14
ship121	19	16	16	17	17
25fv47	32	27	26	25	27

Procedure GTSF $(x^k, p_x, s^k, p_s, f_x, f_s)$

Let

$$lx = \text{argmin}\left\{\frac{x_i}{(p_x)_i} \mid (p_x)_i > 0\right\}$$

and

$$ls = \text{argmin}\left\{\frac{s_i}{(p_s)_i} \mid (p_s)_i > 0\right\}.$$

Compute f_x such that

$$(x^k_{lx} - f_x * (p_x)_{lx})(s^k_{lx} - (p_s)_{lx}) = \frac{(x^k - p_x)^T(s^k - p_s)}{n * \gamma_a}, \tag{6.1}$$

$$f_x := \max(f_x, \gamma_f)$$

and f_s such that

$$(s^k_{lx} - f_s * (p_s)_{ls})(x^k_{ls} - (p_x)_{ls}) = \frac{(x^k - p_x)^T(s^k - p_s)}{n * \gamma_a}, \tag{6.2}$$

$$f_s = \max(f_s, \gamma_f)$$

EXHIBIT 6.1. *Computation of step factor.*

The step factor in the primal space is chosen so that the product of primal blocking variable (lx) and the corresponding dual slack is nearly equal to the value their product would take at the point on the central trajectory at which the duality gap is equal to $(x-p_x)^T(s-p_s)/(n * \gamma_a)$. Note that if $\xi_x = 0$ and $\xi_s = 0$, then $(x-p_x)^T(s-p_s)$ is the duality gap at the point obtained after moving full step. The parameter $\gamma_a > 1$ should be used. The parameter $0 < \gamma_f \leqq 1$ is used to safeguard against very small or negative steps. The explanation for computation of step factor for the dual variables is similar. In essence, the choice of f_x and f_s is guessing the minimizer of potential function (3.5) in directions p_x and p_s while implicitly assuming that $\xi_x = 0$ and $\xi_s = 0$.

Provided that the computations for the search directions were performed with sufficient accuracy, the computational experience on the tested problems shows that the number of iterations required to solve the problems is relatively insensitive to the choice of γ_a and γ_f in a large range. We experimented with values of $\gamma_f = .5, .75, .9, .99, .999$ and $\gamma_a = 1/(1-\gamma_f)$. In our experience we found that on most problems the number of iterations were fewer for a larger choice of γ_f, but the difference was small for γ_f in the range .75 to .999.

However, we observed a very interesting phenomenon. The implementation showed signs of instability for larger values of γ_f for problems brandy, scfxm1, scfxm2, and scfxm3. For these problems to obtain eight digits of accuracy in the solutions at the last two iterations of the algorithm, the conjugate gradient method was needed to improve the accuracy in the search direction. These problems were successfully solved to the desired accuracy for $\gamma_f = .5, .75$ and $\gamma_f = .9$.

An examination of problem data of brandy, scfxm1, scfxm2, and scfxm3 shows that for these problems the set of optimal primal solutions is unbounded. An examination of various stages of our implementation (with different choices of parameters) revealed that allowing for a larger step factor may result in premature convergence of dual slacks corresponding to primal variables unbounded in the optimal set. At a later iteration, this further causes the primal unbounded variables to become very large.

Hence, x_i/s_i corresponding to these variables become disproportionately large. This results in cancellation of large numbers when computing the Cholesky factor, causing computations to become less stable.

The reader is referred to Mehrotra [23] for a more elaborate discussion of the precision of computations in the context of interior point methods and to Mehrotra [22] for a discussion of issues involved in developing implementations based on the preconditioned conjugate gradient method, which we may want to use to improve the numerical accuracy.

7. Initial point. In all of our implementations the initial point is generated as follows. We first compute

$$\tilde{\pi} = (AA^T)^{-1}Ac; \quad \tilde{s} = c - A^T\pi; \quad \tilde{x} = A^T(AA^T)^{-1}b, \tag{7.1}$$

and $\delta_x = \max(-1.5 * \min\{\tilde{x}_i\}, 0)$ and $\delta_s = \max(-1.5 * \min\{\tilde{s}_i\}, 0)$. We then obtain

$$\tilde{\delta}_x = \delta_x + .5 * \frac{(\tilde{x}+\delta_x e)^T(\tilde{s}+\delta_s e)}{\sum_{i=1}^n (\tilde{s}_i + \delta_s)}, \tag{7.2}$$

$$\tilde{\delta}_s = \delta_s + .5 * \frac{(\tilde{x}+\delta_x e)^T(\tilde{s}+\delta_s e)}{\sum_{i=1}^n (\tilde{x}_i + \delta_x)} \tag{7.3}$$

and generate $\pi^0 = \tilde{\pi}$ and $s_i^0 = \tilde{s}_i + \tilde{\delta}_s$, $i = 1, \ldots, n$ and $x_i^0 = \tilde{x}_i + \tilde{\delta}_x$, $i = 1, \ldots, n$ as an initial point.

We first discuss the validity of the above approach in generating $x^0 > 0$ and $s^0 > 0$. In the cases in which it fails to produce such a point, either the problems reduce to that of finding a feasible solution of (P) or (D), or an optimal solution is generated.

From the definition of δ_x and δ_s we know that $x^0 \geqq 0$ and $s^0 \geqq 0$. A positive point is always generated, if $\delta_x > 0$ and $\delta_s > 0$. Furthermore, to show that $x^0 > 0$ and $s^0 > 0$, it is sufficient to show that $\tilde{\delta}_x > 0$ and $\tilde{\delta}_x > 0$.

First consider the case when $\delta_x = 0$ and $\delta_s = 0$. Clearly, in this case $\tilde{x}$ is a feasible solution for (P) and $\tilde{\pi}$, $\tilde{s}$ is a feasible solution for (D). If $\tilde{x}^T\tilde{s} = 0$, then these solutions are optimal for the respective problems. Otherwise, $\tilde{x}^T\tilde{s} > 0$ and hence $\tilde{\delta}_x > 0$ and $\tilde{\delta}_s > 0$. Now consider the case when $\delta_x = 0$ and $\delta_s > 0$. In this case if $\tilde{x}_i \neq 0$ for all i, then obviously $\tilde{\delta}_x > 0$ and $\tilde{\delta}_s > 0$. On the other hand, if $\tilde{x}_i = 0$ for all i, then $b = 0$ and the problem reduces to that of finding a feasible solution of (D). This problem can then be solved separately or by generating a perturbed problem for which the right-hand side is δAe for any positive δ. δe can be used as a feasible interior solution of the perturbed problem, and (7.2)-(7.3) can be used to generate a feasible point of the perturbed problem. Finally, the case when $\delta_x > 0$ and $\delta_s = 0$ can be argued in a similar manner.

We now discuss some properties of the proposed approach.

7.1. Desirable properties of the proposed approach.

Shift in origin. The approach is independent of the shift in origin. To explain what we mean by this, consider the dual problem

$$(D(\Delta)) \qquad \begin{aligned} \text{maximize} \quad & b^T(\pi + \Delta\pi) \\ \text{s.t.} \quad & A^T(\pi + \Delta\pi) + s = c, \\ & s \geqq 0 \end{aligned}$$

for any fixed choice of $\Delta\pi$. Clearly, the polytope defined by the constraints in $(D(\Delta))$ is the same as the polytope defined by the constraints in (D) except for a shift of

origin. It is desirable that an initial point be the same in relation to the respective polytopes. Note that $\tilde{s}$ in (7.1) is independent of the choice of $\Delta\pi$.

To demonstrate how similar arguments hold for (P), we consider an equivalent formulation of this problem. Let Z be a matrix whose columns form the basis for the null space of A, and let x_o be any point satisfying $Ax_o = b$. It is easy to see that (P) is equivalent to

$$(P(\Delta)) \qquad \begin{aligned} \text{minimize} \quad & (c^T Z)(y+\Delta y) \\ \text{s.t.} \quad & Z(y+\Delta y) \geqq -x_o \end{aligned}$$

for $y \in \Re^{n-m}$ and any (fixed) choice of Δy. An approach analogous to that of finding $\tilde{s}$ computes

$$\tilde{y} = -(Z^T Z)^{-1} Z^T x_o.$$

The slacks in the constraint of $(P(Z))$ are given by

$$x_o - Z(Z^T Z)^{-1} Z^T x_o = [I - Z(Z^T Z)^{-1} Z^T] x_o = A^T (AA^T)^{-1} A x_o = A^T (AA^T)^{-1} b = \tilde{x}.$$

Note that $\tilde{x}$ is the orthogonal projection of any vector satisfying $Ax = b$ onto the range space of A. Since $\tilde{x}$ and $\tilde{s}$ are independent of Δy and $\Delta\pi$, respectively, it is obvious that x^0 and s^0 are also independent of this.

Simple scaling. The initial point is not affected if all the constraints in (P) are scaled by a constant or if c is scaled.

7.2. Undesirable properties of the proposed approach.

Column scaling. The initial point is affected by the scaling of columns of A. To illustrate this, in Table 7.1 we give a number of iterations required to solve the problems after problems were scaled by using subroutine M5SCAL from MINOS. All other details of the implementation were kept the same as those for results in Table 8.2.

The results in Table 7.1 indicate that scaling of columns may effect the performance of the implementation. On most problems, use of M5SCAL improved the number of iterations required to solve the problem or the number of iterations did not change significantly. The notable exception was problem fffff800. For this problem, scaling increased the number of iterations by about 40 percent.

We point out that (P) and (D) can be solved in one iteration if we know the correct scaling of columns of A. Hence, the problem of finding the best scaling of columns appears to be as difficult as solving the linear programming problem itself.

Presence of redundant constraint. The initial point generated by using this approach is affected by the presence of redundant constraints. This can be a problem, for example, if redundant dual constraints with large slacks (primal variables with huge costs) are present. In this regard we point out that the central trajectory itself is affected by the presence of such constraints.

A possible way to alleviate this problem would be to ask the user to give relative importance to various primal variables (dual constraints) in solving the problem. A clearly redundant variable could be assigned zero weight, and therefore it could be removed while generating an initial point. In general, the possibility of solving weighted least squares problems to generate initial points in the context of "warm start" should be explored further.

All the approaches [1], [5], [18], [26] used for practical implementations that are reported in the literature are dependent on column scaling and the presence of redundant constraints. Furthermore, they lack one of the desirable properties that the proposed approach has.

TABLE 7.1
Effect of scaling on the performance of the algorithm.

Problem	No scaling	Scaling
afiro	7	6
adlittle	10	10
scagr7	13	14
stochfor1	16	8
sc205	11	11
share2b	12	14
share1b	22	24
scorpion	12	12
scagr25	16	17
sctap1	15	15
brandy	20	17
scsd1	8	7
israel	24	17
bandm	17	15
scfxm1	18	17
e226	20	16
agg	25	16
scrs8	21	18
beaconfd	7	8
scsd6	10	10
ship04s	13	12
agg2	24	20
agg3	21	19
scfxm2	19	20
ship04l	12	12
fffff800	38	52
ship08s	13	14
sctap2	12	12
scfxm3	20	20
ship12s	17	14
scsd8	9	9
czprob	35	30
ship08l	14	15
ship12l	16	15
25fv47	26	23

The choice of constants "1.5" and ".5" in computing $\tilde{\delta}_x$ and $\tilde{\delta}_s$ is arbitrary. The arguments about the validity of the approach (and its properties) do not change if we replace 1.5 with any constant larger than one and .5 with any constant larger than zero. We may do a one-dimensional line search (possibly in direction e) on the potential function (3.5) to generate these constants.

8. Computational performance. The basic ideas presented in this paper were implemented in a FORTRAN code. This section discusses our computational experience with this implementation and compares it with the results documented in the literature.

All testing was performed on a SUN 4/110 work station. The code was compiled with SUN Fortran version 1.0 compiler option "−O3." All the cpu times were obtained by using utility *etime*. The test problems were obtained from *netlib* [7]. The test set includes small and medium size problems. The problem names and additional information on these problems are given in Table 8.1.

All problems were cleaned by using the procedure outlined in Mehrotra [22]. An in-house implementation of the minimum degree heuristic was used to permute the rows. Complete Cholesky factor was computed at each iteration to solve the linear equations. The procedure and the associated data structure, which we used to compute the Cholesky factor, were described in Mehrotra [23]. All the linear algebra subroutines were written by the author.

The algorithm was terminated when the relative duality gap satisfied

$$\frac{c^T x - b^T \pi}{1 + |b^T \pi|} \leqq \varepsilon_{\text{exit}}. \tag{8.1}$$

In the actual implementation ξ_x and ξ_s were computed afresh at each iteration. However, $(\xi_s)_i$ was set to zero and absorbed in s_i if it satisfied $|(\xi_s)_i|/s_i < .001$. This occasionally saved some computational efforts.

The following parameters were set to obtain the results reported in Table 8.2.

$$\begin{aligned}
&\varepsilon_{\text{exit}} = 10^{-8} && \text{(in (8.1))},\\
&\nu = 3 && \text{(in Procedure CENPAR)},\\
&\gamma_f = .9,\ \gamma_a = 10 && \text{(in Procedure GFSF)},\\
&\kappa_x = 100 * \max\{s_i^0\} && \text{(in (3.5))},\\
&\kappa_s = 100 * \max\{x_i^0\} && \text{(in (3.5))}.
\end{aligned}$$

The number of iterations required to solve the problems is given in the second column of Table 8.2. The primal objective value recorded at termination is given in column 3. The relative duality gap (8.1) is given in column 4. The primal infeasibility,

$$\|Ax - b\|/(1 + \|x\|), \tag{8.2}$$

and the dual infeasibility,

$$\|A^T \pi + s - c\|/(1 + \|s\|), \tag{8.3}$$

recorded at termination are given in columns 5 and 6, respectively. This information on relative duality gap and primal and dual feasibility is the same as that given in Lustig, Marsten, and Shanno [18]. We use the same stopping criterion and provide similar information in order to be consistent while making comparisons.

All the problems were accurately solved to eight digits. A comparison with the results in Lustig, Marsten, and Shanno [18] show that on many problems in our implementation the accuracy in the objective value at termination was better. This is primarily due to our approach for computing the step factor.

In Table 8.3 we compare the number of iterations required to solve the test problems. Column 2 of this table gives the number of iterations taken by our implementation. Column 3 gives the number of iterations taken by the dual affine

scaling method, as reported by Adler et al. [1]. Column 4 gives the number of iterations required by the second-order dual affine scaling method in Adler et al. [1]. Column 5 gives the number of iterations reported in Lustig, Marsten, and Shanno [18] for a primal-dual method. Column 6 gives the number of iterations reported in Gill, Murray, and Saunders [8] for a logarithmic barrier function method. Column 7 gives the number

TABLE 8.1
Problem statistics.

Problem	Rows	Columns	Nonzeros
afiro	28	32	88
adlittle	57	97	465
scagr7	130	140	553
stochfor1	118	111	474
sc205	206	203	552
share2b	97	79	730
share1b	118	225	1,182
scorpion	389	358	1,708
scagr25	472	500	2,029
sctap1	301	480	2,052
brandy	221	249	2,150
scsd1	78	760	3,148
israel	175	142	2,358
bandm	306	472	2,659
scfxm1	331	457	2,612
e226	224	282	2,767
agg	489	163	2,541
scrs8	491	1,169	4,029
beaconfd	174	262	3,476
scsd6	148	1,350	5,666
ship04s	403	1,458	5,810
agg2	517	302	4,515
agg3	517	302	4,531
scfxm2	661	914	5,229
ship04l	403	2,118	8,450
fffff800	525	854	6,235
ship08s	779	2,387	9,501
sctap2	1,091	1,880	8,124
scfxm3	991	1,371	7,846
ship12s	1,152	2,763	10,941
scsd8	398	2,750	11,334
czprob	930	3,523	14,173
ship08l	779	4,283	17,085
ship12l	1,152	5,427	21,597
25fv47	822	1,571	11,127

TABLE 8.2
Computational performance of the implemented algorithm.

Problem	itn	$c^T x^k$	$b^T \pi^k$	$\frac{\lvert c^T x^k - b^T \pi^k \rvert}{1+\lvert b^T \pi^k \rvert}$	$\frac{\lVert Ax^k - b \rVert}{1+\lVert x^k \rVert}$	$\frac{\lVert A^T \pi^k + s^k - c \rVert}{1+\lVert s^k \rvert}$	$\frac{\mu^k}{1+\lvert b^T \pi^k \rvert}$
afiro	7	−464.75314272968	−464.75314309228	7e−10	5e−15	0	4e−13
adlittle	10	225,494.96330271	225,494.96311956	8e−10	1e−11	5e−14	4e−13
scagr7	13	−2,331,389.8242996	−2,331,389.8243996	4e−11	2e−13	0	2e−13
stochfor	16	−41,131.976219436	−41,131.976219346	3e−15	4e−15	0	3e−26
sc205	11	−52.202061191133	−52.202061364254	3e−9	4e−12	0	7e−14
share2b	12	−415.73224070502	−415.73224074419	9e−11	8e−12	8e−18	1e−16
share1b	22	−76,589.318579203	−76,589.318579231	3e−13	5e−13	0	1e−15
scorpion	12	1,878.1248227381	1,878.1248227369	5e−13	6e−14	7e−16	1e−17
scagr25	16	−14,753,443.031527	−14,753,433.097009	4e−9	1e−11	1e−16	5e−12
sctap1	15	1,412.2500000000	1,412.2500000000	1e−14	2e−14	0	9e−28
brandy	20	1,518.5098972369	1,518.5098964881	4e−10	1e−09	1e−17	9e−20
scsd1	8	8.6666666743334	8.6666666743334	7e−15	5e−15	0	2e−27
israel	24	−896,644.82032330	−896,644.82213083	2e−10	5e−14	0	9e−14
bandm	17	−158.62801845009	−158.62801845012	2e−13	3e−14	1e−16	3e−18
scfxm1	18	18,416.759026662	18,416.759028090	7e−11	2e−10	4e−16	2e−14
e226	20	−18.751929066371	−18.751929066371	1e−14	6e−14	2e−17	1e−20
agg	25	−35,991,767.286586	−35,991,767.286789	5e−12	2e−13	0	8e−11
scrs8	21	904.29695380418	904.29695377855	2e−11	1e−15	3e−17	2e−13

TABLE 8.2 (*continued*).

Problem	itn	$c^T x^k$	$b^T \pi^k$	$\frac{\lvert c^T x^k - b^T \pi^k \rvert}{1+\lvert b^T \pi^k \rvert}$	$\frac{\lVert Ax^k - b \rVert}{1+\lVert x^k \rVert}$	$\frac{\lVert A^T \pi^k + s^k - c \rVert}{1+\lVert s^k \rVert}$	$\frac{\mu^k}{1+\lvert b^T \pi^k \rvert}$
beaconfd	7	33,592.485814826	33,592.485771448	1e−9	2e−10	2e−11	2e−11
scsd6	10	50.500000078275	50.500000064030	2e−10	6e−15	0	2e−19
ship04s	13	1,798,714.7004927	1,798,714.7004444	2e−11	6e−13	1e−16	1e−12
agg2	24	−20,239,252.354277	−20,239,252.356863	1e−10	5e−14	0	1e−11
agg3	21	10,312,115.9350899	10,312,115.9350813	8e−13	2e−14	0	1e−10
scfxm2	19	36,660.261567320	36,660.261564919	6e−11	1e−10	3e−16	1e−11
ship04l	12	1,793,324.5379701	1,793,324.5379704	1e−13	1e−13	2e−16	3e−14
fffff800	38	555,679.56576888	555,679.56340175	4e−9	2e−10	0	2e−11
ship08s	13	1,920,098.2105549	1,920,098.2105340	1e−11	1e−13	0	3e−13
sctap2	12	1,724.8071428587	1,724.8071428563	1e−12	1e−14	0	7e−16
scfxm3	20	54,901.254586753	54,901.254517497	1e−9	2e−10	6e−15	4e−13
ship12s	16	1,489,236.1347549	1,489,236.1338498	1e−11	3e−12	1e−13	1e−15
scsd8	9	905.00000451087	904.99999866657	6e−9	3e−15	0	2e−15
czprob	35	2,185,196.7039679	2,185,196.6964335	3e−9	2e−12	0	1e−14
ship08l	14	1,909,055.2113734	1,909,055.2113891	8e−12	1e−11	9e−17	6e−20
ship12l	16	1,470,187.9194253	1,470,187.9193126	7e−11	2e−13	6e−15	3e−18
25fv47	26	5,501.8458883209	5,501.8458882282	7e−11	1e−14	2e−17	1e−17

TABLE 8.3
Comparison of number of iterations with other implementations.

Problem	AIPM	AKRV2	AKRV1	LMS	GMS	DBDW
afiro	7	15	20	13	20	10
adlittle	10	18	24	17	18	15
scagr7	13	19	24	22	24	18
stochfor1	16			19		
sc205	11	20	28	16	32	17
share2b	12	21	29	17	46	14
share1b	22	33	38	40	35	28
scorpion	12	19	24	18	33	17
scagr25	16	21	29	24	28	21
sctap1	15	23	33	22	53	21
brandy	20	24	38	27	41	21
scsd1	8	16	19	12	13	8
israel	24	29	37	47	36	24
bandm	17	24	30	28	31	21
scfxm1	18	30	33	31	37	23
e226	20	30	34	31	38	24
agg	25			32		
scrs8	21	29	39	50	59	25
beaconfd	7	17	23	21	34	17
scsd6	10	18	22	15	15	13
ship04s	13	22	30	21	40	15
agg2	24			32		
agg3	21			32		
scfxm2	19	29	39	37	42	29
ship04l	12	21	28	22	36	17
fffff800	38			59	55	
ship08s	13	21	32	23	34	16
sctap2	12	25	34	23	41	16
scfxm3	20	30	40	39	42	31
ship12s	16	23	35	27	46	16
scsd8	9	18	23	15	15	13
czprob	35	35	52	57	56	41
ship08l	14	23	31	24	22	17
ship12l	16	23	32	27	24	17
25fv47	24		52	48	44	28

of iterations reported in Domich et al. [5] for a variant of the method of centers. Our method and the second-order dual affine scaling method in Adler et al. [1] computes two directions at each iteration. The method implemented in Domich et al. [5] computes three directions at each iteration and solves a linear programming problem defined by

using these directions. All other implementations compute only one direction at each iteration.

On the average the number of iterations required by our implementation to solve the mutually tested problems is 40 percent less than that reported in Lustig, Marsten, and Shanno [18]; it is 50 percent less than that reported in Adler et al. [1]; and it is 55 percent less than that reported by Gill, Murray, and Saunders [8]. Compared to the results in Adler et al. [1] for their second-order method, the results show that our method takes about 35 percent fewer iterations. Finally, our implementation required 20 percent fewer iterations than those required in Domich et al. [5].

It is useful to point out that it is possible to further reduce the total number of iterations needed to solve the problems by using higher-order derivatives. This is discussed in a subsequent paper.

The number of iterations was always fewer than the number of iterations required by the second-order dual affine scaling method implemented by Adler et al. [1]. Many of the problems were solved in practically half the number of iterations when compared with [1].

9. Comparison with OB1 and MINOS 5.3. This section compares the cpu times required by our implementation to those required by the implementation of the primal-dual method in OB1 [18] (02/90 version) and the simplex method in MINOS 5.3 [29]. The source codes (also written in FORTRAN) of OB1 (02/90 version) and MINOS 5.3 were compiled using compiler option "−O3." Hence, everything was identical while making these comparisons. All the default options of OB1 (02/90 version) and MINOS 5.3 were used. Printing was turned to minimum level in both cases. In the case of OB1 (02/90 version), crush = 2 was used for all problems.

The times for MINOS 5.3 are those for subroutine M5SOLV only. The TIMER subroutine in OB1 was used to compute its cpu times. The times for OB1 were calculated as follows:

$$\text{OB1 Time} = \text{end of hprep} - \text{after mpsink} + \text{end of obdriv} - \text{after getcmo}.$$

Times required by MINOS 5.3, OB1 (02/90 version), and our implementation do not include times spent in converting the MPS input file into a problem in the standard form. The times required by our implementation include all the time spent after the input files were converted into a problem in the standard form.

The times required by our implementation is given in the second column of Table 9.1. The times required by OB1 (02/90 version) are given in column 3 and the times required by MINOS are given in column 4. Column 5 gives the ratio of times required by OB1 (02/90 version) to our code. Column 6 gives the ratio of times required by MINOS 5.3 to our code.

From these results we find that, on the average, our implementation in the current state performs two times better than the implementation in OB1 (02/90 version). The ratio of cpu times with OB1 (02/90 version) is more or less uniform.

Comparing the results with MINOS 5.3 we find that the proposed implementation is on the average better by a factor of 2.5. In this case, however, the ratio of cpu times varies significantly. MINOS 5.3 was generally superior on problems with few relatively dense columns, whereas our implementation of the primal-dual method was superior on problems with sparse Cholesky factor.

10. Conclusions. Details of a particular implementation of the primal-dual method are given. This implementation requires a considerably smaller number of iterations and saves considerable computational effort. We have given expressions to compute

TABLE 9.1
Comparison of cpu *time with* OB1 *and* MINOS 5.3 *on* SUN 4/110.

Problem	AIPM	OB1	MINOS5.3	$\frac{\text{OB1}}{\text{AIPM}}$	$\frac{\text{MINOS5.3}}{\text{AIPM}}$
afiro	.12	.60	.09	5.0	.7
adlittle	.64	1.81	.70	2.8	1.1
scagr7	1.11	2.75	1.66	2.5	1.5
stochfor1	1.48	2.91	1.50	2.0	1.0
sc205	1.49	3.33	2.16	2.2	1.4
share2b	1.50	3.00	1.46	2.0	1.0
share1b	3.27	9.21	3.98	2.8	1.2
scorpion	2.87	7.06	5.92	2.4	2.0
scagr25	5.23	10.43	15.32	2.0	2.9
sctap1	4.92	9.18	7.71	1.8	1.6
brandy	7.18	15.30	11.55	2.1	1.6
scsd1	2.46	5.46	6.15	2.2	2.5
israel	58.37	127.01	6.11	2.2	.1
bandm	8.01	17.61	22.09	2.2	2.7
scfxm1	10.55	20.82	12.76	2.0	1.2
e226	9.38	15.83	15.30	1.7	1.6
agg	32.88	47.46	7.32	1.4	.2
scrs8	13.31	43.95	40.86	3.3	3.1
beaconfd	2.56	9.28	1.97	3.6	.8
scsd6	5.66	10.56	31.19	1.9	5.5
ship04s	6.92	17.58	6.63	2.5	.95
agg2	65.86	100.92	10.05	1.5	.15
agg3	66.66	94.74	10.95	1.4	0.16
scfxm2	21.69	46.74	51.42	2.1	2.37
ship04l	8.90	24.35	13.20	2.7	.54
fffff800	80.90	140.01	14.33	1.7	.17
ship08s	9.50	23.05	20.43	2.4	2.1
sctap2	30.04	47.75	56.02	1.6	1.9
scfxm3	33.31	72.84	107.40	2.1	3.2
ship12s	14.06	31.78	47.85	2.2	3.4
scsd8	10.38	21.98	230.23	2.1	22.2
czprob	33.78	81.93	166.03	2.4	4.9
ship08l	18.12	42.81	19.34	2.4	1.0
ship12l	28.56	63.11	120.31	2.2	4.2
25fv47	164.53	334.14	941.41	2.0	5.7
Total	766.20	1,507.29	2,011.4		

all the derivatives at a given point of a primal-dual affine scaling trajectory. The implementation described here effectively combines the second derivative with the centering vector. Heuristics for computing centering parameter and step length were given and their effectiveness was demonstrated. A new approach to generating a starting point was used. In addition, the results demonstrate that it is possible to develop fast (robust) implementations of interior point methods, which ensure sufficient reduction in a potential function at each iteration.

Comparison with OB1 (02/90 version) and the simplex method show that our implementation was faster by a factor of 2 and 2.5, respectively.

Acknowledgments. I thank Professor Michael Saunders for making the source code of MINOS 5.3 available. I thank Professor Roy E. Marsten for releasing a copy of OB1 for comparison in this paper. Also, I thank Mr. I. C. Choi for helping out with OB1 and Professor Donald Goldfarb for his constant encouragement, which made this work possible. I also thank the two anonymous referees for their careful reading of this paper and for their suggestions for improvement.

Appendix. Here we prove that the potential function (3.3) can be reduced by a constant amount at each iteration. The development of our proof is based on the analysis in Freund [6]. Let us define $l \equiv 2n+1$,

$$\hat{A} \equiv \begin{bmatrix} A & 0 & -\xi_x \\ 0 & Z^T & -\xi_s \\ c^T & (AA^T)^{-1}A & \xi_a \end{bmatrix},$$

$\hat{b}^T \equiv (b^T, c^T Z, b^T(AA^T)^{-1}Ac)$, and $y = (x^T, s^T, \lambda)^T$. Hence, without loss of generality, consider the problem

$$(PD) \qquad \begin{aligned} \text{minimize} \quad & y_l \\ \text{s.t.} \quad & \hat{A}y = \hat{b}, \\ & y \geqq 0, \end{aligned}$$

where $y \in \Re^l$. Let us consider the potential function

$$F(y) \equiv \hat{\rho} \ln y_l - \sum_{i=1}^{l} \ln y_i, \tag{A.1}$$

where $\hat{\rho} = l + \sqrt{l}$. The function $F(y)$ in (A.1) is the same as the function (3.3). Let $y^k > 0$ be any feasible point of (PD) and let Y^k be the diagonal matrix whose diagonal elements are $(y_1, \ldots, y_l)$. Let

$$d \equiv [I - \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{A}](\hat{\rho}e_l - e),$$

where $\bar{A} = \hat{A}Y^k$. Let $y^{k+1} = y^k - (\varepsilon/\|d\|)Y^k d$, $\varepsilon < 1$. Then

$$\begin{aligned} F(y^{k+1}) - F(y^k) &= \hat{\rho} \ln \frac{y_l^{k+1}}{y_l^k} - \sum_{i=1}^{l} \ln \frac{y_i^{k+1}}{y_i^k} \\ &= \hat{\rho} \ln \left(1 - \frac{\varepsilon}{\|d\|} d_l\right) - \sum_{i=1}^{l} \ln \left(1 - \frac{\varepsilon}{\|d\|} d_i\right) \\ &\leqq -\frac{\hat{\rho}\varepsilon}{\|d\|} d_l + \frac{\varepsilon}{\|d\|} \sum_{i=1}^{n} d_i + \frac{\varepsilon^2}{2(1-\varepsilon)} \\ &= -\varepsilon\|d\| + \frac{\varepsilon^2}{2(1-\varepsilon)}. \end{aligned}$$

The inequality above follows by using the fact that $\ln(1+\delta) \leqq \delta$ for $\delta > -1$, and $\ln(1+\delta) \geqq (\delta^2/2(1-\varepsilon))$ if $|\delta| \leqq \varepsilon < 1$.

If $\|d\| \geqq 1$, then for $\varepsilon = .5$, $F(y)$ is reduced by .25.

Otherwise, if $\|d\| < 1$, then from the definition of d, we have

$$\bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{A}\left(y_l^k e_l - \frac{y_l^k}{\hat{\rho}} e\right) + \frac{y_l^k}{\hat{\rho}}(d+e) = y_l^k e_l,$$

which gives a feasible solution to the dual of (PD). Furthermore, the duality gap is given by

$$y_l^k \frac{e^T(d+e)}{\hat{\rho}} \leqq y_l^k \frac{l+\sqrt{l}\|d\|}{l+\sqrt{l}} < y_l^k.$$

But y_l^k is also the current objective value. Therefore, the optimal objective value of (PD) must be positive. If this is the case, then either (P) or (D) has no feasible solution, and we would stop.

Hence, if (P) and (D) have a feasible solution then $F(y)$ can be reduced by .25 at each iteration. A failure to reduce $F(y)$ by this amount would imply that either (P) or (D) has no feasible solution.

REFERENCES

[1] I. ADLER, N. KARMARKAR, M. G. C. RESENDE, G. VEIGA (1989), *An implementation of Karmarkar's algorithm for linear programming*, Math. Programming, 44, pp. 297–336.

[2] D. A. BAYER AND J. C. LAGARIAS (1989), *The nonlinear geometry of linear programming:* I. *Affine and projective scaling trajectories*, II. *Legendre transform coordinates and central trajectories*, Trans. Amer. Math. Soc., pp. 499–581.

[3] Y. C. CHENG, D. J. HOUCK, J. M. LIU, M. S. MEKETON, L. SLUTSMAN, R. J. VANDERBEI, AND P. WANG (1989), *The* AT&T KORBX *System*, AT&T Tech. J., 68, pp. 7–19.

[4] I. C. CHOI, C. MONMA, AND D. F. SHANNO (1990), *Further development of a primal-dual interior point method*, ORSA J. Comput., 2, pp. 304–311.

[5] P. D. DOMICH, P. T. BOGGS, J. R. DONALDSON, AND C. WITZGALL (1989), *Optimal 3-dimensional methods for linear programming*, NISTIR 89-4225, Center for Computing and Applied Mathematics, U.S. Dept. of Commerce, National Institute of Standards and Technology, Gaithersburg, MD.

[6] R. FREUND (1988), *Polynomial-time algorithms for linear programming based only on primal scaling and projected gradient of a potential function*, Working Paper OR 182-88, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA.

[7] D. M. GAY (1985), *Electronic mail distribution of linear programming test problems*, Mathematical Programming Society Committee on Algorithms News Letter, 13, pp. 10–12.

[8] P. E. GILL, W. MURRAY, AND M. A. SAUNDERS (1988), *A single-phase dual barrier method for linear programming*, Report SOL 88-10, Systems Optimization Laboratory, Stanford Univ., Stanford, CA.

[9] D. GOLDFARB AND S. MEHROTRA (1988), *A relaxed of Karmarkar's algorithm*, Math. Programming, 40, pp. 285–315.

[10] ——— (1989), *A self-correcting version of Karmarkar's algorithm*, SIAM J. Numer. Anal., 26, pp. 1006–1015.

[11] C. GONZAGA AND M. J. TODD (1989), *An $O(\sqrt{n}L)$ iterations large-step primal-dual affine algorithm for linear programming*, Tech. Report, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY.

[12] D. D. HERTOG AND C. ROSS (1989), *A survey of search directions in interior point methods for linear programming*, Report 89-65, Faculty of Mathematics and Informatics/Computer Science, Delft Univ. of Technology, Delft, Holland.

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

[14] N. KARMARKAR, J. C. LAGARIAS, L. SLUTSMAN, AND P. WANG (1989), *Power series variants of Karmarkar-type algorithms*, AT&T Tech. J., May/June, pp. 20–36.

[15] M. KOJIMA, S. MIZUNO, AND A. YOSHISE (1989), *A primal-dual interior point algorithm for linear programming*, in Progress in Mathematical Programming, Interior Point and Related Methods, N. Megiddo, ed., Springer-Verlag, New York, pp. 29–47.

[16] ——— (1991), *An $O(\sqrt{n}L)$-iteration potential reduction algorithm for linear complementarity problems*, Math. Programming, 50, pp. 331–342.

[17] I. J. LUSTIG (1991), *Feasibility issues in a primal-dual interior-method for linear programming*, Math. Programming, 49, pp. 145–162.

[18] I. J. LUSTIG, R. E. MARSTEN, AND D. F. SHANNO (1989), *Computational experience with a primal-dual interior point method for linear programming*, TR J-89-11, Industrial and Systems Engineering Report Series, Georgia Inst. of Technology, Atlanta, GA.

[19] R. E. MARSTEN, M. J. SALTZMAN, D. F. SHANNO, G. S. PIERCE, AND J. F. BALLINTIJN (1989), *Implementation of a dual affine interior point algorithm for linear programming*, ORSA J. Comput. 1, pp. 287–297.

[20] K. A. MCSHANE, C. L. MONMA, AND D. SHANNO (1989), *An implementation of a primal-dual interior point method for linear programming*, ORSA J. Comput., 1, pp. 70–83.

[21] N. MEGIDDO (1986), *Pathways to the optimal set in linear programming*, in Progress in Mathematical Programming, N. Megiddo, ed., Springer-Verlag, New York, pp. 131–158.

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

[23] ——— (1989), *Implementations of affine scaling methods: Towards faster implementations with complete Cholesky factor in use*, TR-89-15R, Dept. of Industrial Engineering/Management Science, Northwestern Univ., Evanston, IL.

[24] ——— (1991), *On finding a vertex solution using interior point methods*, Linear Algebra Appl., 152, pp. 233–253.

[25] ——— (1990), *On an implementation of primal-dual predictor-corrector algorithms*, presented at the Second Asilomar Workshop on Progress in Mathematical Programming, Asilomar, CA.

[26] C. L. MONMA AND A. J. MORTON (1987), *Computational experience with a dual affine variant of Karmarkar's method for linear programming*, Oper. Res. Lett., 6, pp. 261–267.

[27] R. C. MONTEIRO AND I. ADLER (1989), *An $O(n^3L)$ primal-dual interior point algorithm for linear programming*, Math. Programming, 44, pp. 43–66.

[28] R. C. MONTEIRO, I. ADLER, AND M. G. C. RESENDE (1990), *A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension*, Math. Oper. Res., 15, pp. 191–214.

[29] B. A. MURTAGH AND M. A. SAUNDERS (1983), MINOS 5.0 *user's guide*, Tech. Rep. SOL 83-20, Dept. of Operations Research, Stanford Univ., Stanford, CA.

[30] M. J. TODD (1989), *The affine-scaling direction for linear programming is a limit of projective-scaling direction*, Tech. Rep., School of Operations Research and Industrial Engineering, Cornell Univ., Ithaca, NY.

[31] M. J. TODD AND Y. YE (1990), *A centered projective algorithm for linear programming*, Math. Oper. Res., 15, pp. 508–529.

[32] Y. YE (1991), *An $O(n^3L)$ potential reduction algorithm for linear programming*, Math. Programming, 50, pp. 239–258.