Higher Order Methods and their Performance

Sanjay Mehrotra[†]

Department of Industrial Engineering and Management Sciences,
Northwestern University, Evanston, IL 60208-3119, U.S.A.

Technical Report 90-16R1

last revision: July 1991

Abstract

A class of primal-dual interior point methods is developed. These methods use a higher order Taylor polynomial to approximate a primal-dual trajectory defined from an infeasible point. A detailed implementation of these methods is given and their computational performance is studied.

We give computational results for two different approaches for defining trajectory, and using up to $10th$ order polynomial. In all cases, all the tested **netlib** problems were solved using default settings. The use of higher order polynomial on the average results in approximately 25% to 35% reduction in the number of iterations. On several difficult problems in the **netlib** test set this translates into 10% to 15% savings in *cpu* time.

[†] Supported in part by a grant from the GTE Laboratories and by grants CCR-88-10107, and CCR-9019469 from the National Science Foundation.

Introduction

This paper is on the development and computational performance of higher order methods for solving linear programs. These methods compute first and higher derivatives of an appropriately defined trajectory. This trajectory starts from any positive point and goes through an optimal solution of the problem. A Taylor polynomial is constructed from these derivatives and used to update the current solution. The methods developed here build upon the work of Mehrotra [13].

The motivation for developing higher order methods comes from the observation that the most expensive step at an iteration of an interior point method is the factorization of a matrix, which changes numerically at each iteration. A first order method (e.g., Monterio and Adler [17], Kojima, Mizuno and Yoshise [10], Lustig, Marsten and Shanno [11]) uses the first derivative (Newton direction) of some trajectory and uses it as a search direction to move to a new point. An important observation (e.g., see [8, 18]) is that higher derivatives of a trajectory can be computed recursively by using the same factors used to compute the first derivative. Therefore, if Gaussian elimination is used for a dense problem, while the computation of factors and the first derivative requires $O(n^3)$ arithmetic operations, the computation of an additional derivative requires only $O(n^2)$ arithmetic operations. If the use of additional derivatives reduces the number of iterations significantly, then its computation would be well worth it.

The use of higher order information was proposed by Karmarkar, Lagarias, Slutsman and Wang [8], and Megiddo [16]. Karmarkar *et al.* had given a class of methods based on a "reparameterized" trajectory motivated from the use of differential equations. These methods are implemented in the AT&T KORBX system [4]. Monterio, Adler and Resende [18] proved some interesting results for a higher order primal-dual affine scaling method by closely following the central trajectory. The use of higher order information is also considered by Domich, Boggs, Donaldson and Witzgall [5] in the method of centers. In their method a linear program in a three dimensional subspace is solved to generate a search direction.

The methods we develop in this paper differ from the methods in [8] in several ways. The approach we use to construct trajectories is different from the one given in [8]. The construction in [8] requires knowledge of feasible solutions. In our case, we need not start from a feasible point. Furthermore, we do not reparameterize our trajectories while taking approximations.

In this paper we give two specific approaches for defining a trajectory, using a general principle. The practical performance of algorithms based on these approaches is studied and computational results are given. The results we present here used up to $10th$ order polynomial for both algorithms. All the tested problems from **netlib**[7] were solved to desired accuracy using the default setting in all the cases.

The use of higher derivatives result in a significant reduction in the number of iterations required to solve the problems. For example the use of up to $10th$ order polynomial results in approximately 25% to 35% reduction over the second order polynomial and approximately 50% to 70% over the first order methods. On several *netlib* problems for which factorization of matrix required for computing a search direction is expensive, this results in significant savings in the *cpu* times. Interestingly, empirical evidence suggests that the reduction in the number of iterations diminishes with increasing order of polynomial.

This paper describes a part of our on going research on interior point methods. We have tried to make this paper self contained, however, it is important to point out that the work described in Mehrotra[14] and Fourer and Mehrotra[6] has contributed significantly to the numerical stability

of the implementations of algorithms described here.

This paper is organized as follows. In Section 1 we give our general approach for generating trajectories. This section also develops expressions for recursively computing all derivatives of a trajectory. In Section 2 we give details of two higher order methods as we implemented them. Finally, Section 3 discusses computational results obtained on the **netlib** test set.

1. A Class of Trajectories and their Derivatives

Let us consider the linear programming problem (P) :

$$\begin{array}{rl} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \\ & x + y = d \\ & x, y \geq 0, \end{array}$$

where $c, x, y, d \in \Re^n$, $b \in \Re^m$, and $A \in \Re^{m \times n}$. We assume that A has full row rank. This can be ensured by removing linearly dependent rows in the beginning. The second set of constraints in (P) are for upper bounds on variables. These constraints are introduced to develop expressions for problems with bounded variables. We assume that all variables have an upper bound to simplify our notations. In practice we may also have free variables. We refer the reader to Mehrotra[14] for our treatment of free variables. The expressions developed here extend to that case in a straight forward way.

The dual of (P) is given by (D) :

$$\begin{array}{rl} \text{maximize} & b^T \pi - d^T s \\ \text{subject to} & A^T \pi + r - s = c, \\ & r, s \geq 0. \end{array}$$

From the optimality conditions for linear programming a point $\Gamma^* = (x^*, y^*, \pi^*, r^*, s^*)$ is optimal if it satisfies (PD) :

$$\begin{array}{rl} Xr & = 0, \\ Ys & = 0, \\ Ax & = b, \\ x + y & = d, \\ A^T \pi + r - s & = c, \\ x \geq 0,\ y \geq 0, & \quad r \geq 0,\ s \geq 0. \end{array}$$

Here X represents a diagonal matrix with $X_{ii} = x_i$. Similar notation is used for other variables.

Assume that $x^{(0)} > 0,\ y^{(0)} > 0,\ \pi^{(0)},\ r^{(0)} > 0,\ s^{(0)} > 0$ is a given point. Let $\xi_b = Ax^{(0)} - b$, $\xi_d = x^{(0)} + y^{(0)} - d$, and $\xi_c = A^T \pi^{(0)} + r^{(0)} - s^{(0)} - c$. ξ_b, ξ_d and ξ_c are the residuals in the primal and dual constraints at the current point. Let

$$f(\alpha) = \begin{cases} 0, & \alpha = 0, 1 \\ > 0, & \alpha \in (0, 1) \end{cases} \tag{1.1}$$

and

$$g(\alpha) = \begin{cases} 1, & \alpha = 1 \\ 0, & \alpha = 0 \\ \in (0, 1), & \alpha \in (0, 1). \end{cases} \tag{1.2}$$

The functions $f(\alpha)$ and $g(\alpha)$ are used to construct trajectories that guide us to a solution of (PD).

To develop these trajectories consider the system of equations:

$$\left.\begin{array}{rl} X(\alpha)r(\alpha) & = g(\alpha)X^{(0)}r^{(0)} + f(\alpha)\mu e, \\ Y(\alpha)s(\alpha) & = g(\alpha)Y^{(0)}s^{(0)} + f(\alpha)\mu e, \\ Ax(\alpha) & = b + g(\alpha)\xi_b, \\ x(\alpha) + y(\alpha) & = d + g(\alpha)\xi_d, \\ A^T\pi(\alpha) + r(\alpha) - s(\alpha) & = c + g(\alpha)\xi_c, \\ x(\alpha) \geq 0,\ r(\alpha) \geq 0, & \ s(\alpha) \geq 0 \end{array}\right\} \tag{1.3}$$

for $\alpha \in [0,1]$. Here e is a vector of all ones. If we let $x(1) \equiv x^{(0)}$, $y(1) \equiv y^{(0)}$, $\pi(1) \equiv \pi^{(0)}$, $r(1) \equiv r^{(0)}$, $s(1) \equiv s^{(0)}$, then (1.3) are satisfied by the current point for $\alpha = 1$.

The following theorem is about the existence of solutions of (1.3). It is a generalization of the Proposition 4.1 in Mehrotra[13].

Theorem 1.1: *If (PD) has a solution, then (1.3) has a solution for all $\alpha \in [0,1]$. Furthermore, this solution is unique for any $\alpha \in (0,1]$.*

Proof: Let $v_i(\alpha) = g(\alpha)x_i^{(0)}r_i^{(0)} + f(\alpha)\mu$ and $w_i(\alpha) = g(\alpha)y_i^{(0)}s_i^{(0)} + f(\alpha)\mu$. From (1.1), (1.2) and the assumptions on $x^{(0)}, y^{(0)}, r^{(0)}, s^{(0)}$ it is easy to see that $v_i(\alpha) > 0$ and $w_i(\alpha) > 0$ for all i. For a fixed α (1.3) are also the optimality conditions for a weighted logarithmic barrier problems (P_α) :

$$\begin{array}{rl} \text{minimize } B(x,y,\alpha) \equiv & (c + g(\alpha)\xi_c)^T x - \sum_{i=1}^n v_i(\alpha)\ln x_i - \sum_{i=1}^n w_i(\alpha)\ln y_i \\ \text{subject to} & Ax = b + g(\alpha)\xi_b, \\ & x + y = d + g(\alpha)\xi_d \\ & x, y > 0, \end{array}$$

and (D_α) :

$$\begin{array}{rl} \text{maximize} & (b + g(\alpha)\xi_b)^T\pi - (d + g(\alpha)\xi_d)^T s + \sum_{i=1}^n v_i(\alpha)\ln r_i + \sum_{i=1}^n w_i(\alpha)\ln s_i \\ \text{subject to} & A^T\pi + r - s = c + g(\alpha)\xi_c, \\ & r, s > 0. \end{array}$$

Let $x(0), y(0), \pi(0), r(0), s(0)$ represent a solution of (1.3) for $\alpha = 0$. For a given $\alpha \in (0,1)$, since $g(\alpha) \in (0,1)$, $\tilde{x}(\alpha) = (1 - g(\alpha))x(0) + g(\alpha)x^{(0)}$, $\tilde{y}(\alpha) = (1 - g(\alpha))y(0) + g(\alpha)y^{(0)}$ is a feasible solution for (P_α) and $\tilde{\pi}(\alpha) = (1 - g(\alpha))\pi(0) + g(\alpha)\pi^{(0)}$, $\tilde{r}(\alpha) = (1 - g(\alpha))r(0) + g(\alpha)r^{(0)}$, $\tilde{s}(\alpha) = (1 - g(\alpha))s(0) + g(\alpha)s^{(0)}$ is a feasible solution for (D_α). If the feasible set of (P_α) is bounded, then obviously (P_α) has a solution. Furthermore, this solution is unique because $B(x,y,\alpha)$ is a strictly convex function.

Now note that the feasible set of (P_α) is bounded because of the bound constraints. This completes the proof for the problem form in consideration. However, since in general all variables would not have upper bounds, for Theorem 1.1 to be useful, it should hold for the case where the feasible set of (P_α) is unbounded. In fact, this is true. A proof can be given for this by following the arguments in the proof of Proposition 4.1 in Mehrotra[13]. We omit it here. □

The general form of (1.3) gives enormous freedom in the way a trajectory can be defined. We now comment on the conditions imposed on $f(\alpha)$ and $g(\alpha)$ in (1.1) and (1.2) respectively.

The requirement that $f(\alpha)$ and $g(\alpha)$ be non-negative ensures that the defined trajectory stays in the positive orthant. The requirement that $g(\alpha) = 1$ for $\alpha = 1$ and $g(\alpha) = 0$ for $\alpha = 0$ and $f(\alpha) = 0$ for $\alpha = 1, 0$ ensures that the trajectory starts from the current point and goes through

an optimal solution. Because of the restriction $f(\alpha) = 0$ for $\alpha = 0$ we need not assume that the problem has a primal and dual interior feasible solution to define a trajectory. This is important because many real problems in their original formulation do not have an interior solution.

If a problem is known to have an interior solution then conditions (1.1) can be replaced with

$$f(\alpha) = \begin{cases} 0, & \alpha = 1 \\ 1, & \alpha = 0 \\ > 0, & \alpha \in (0,1) \end{cases}$$

In this case the defined trajectory would start from the current point and go through a point on the central trajectory.

We now show that all the derivatives of (1.3) can be computed recursively provided that $f(\alpha)$ and $g(\alpha)$ are infinitely differentiable. Let $\Gamma(\alpha) = (x(\alpha), y(\alpha), \pi(\alpha), r(\alpha), s(\alpha))$ represent the solution of (1.3) for some α. Let $\Gamma^{(l)}(\alpha)$ represent the lth derivative of $\Gamma(\alpha)$ with respect to α. Similarly let $x^{(l)}(\alpha)$, $y^{(l)}(\alpha)$, $\pi^{(l)}(\alpha)$, $r^{(l)}(\alpha)$, $s^{(l)}(\alpha)$, $f^{(l)}(\alpha)$, $g^{(l)}(\alpha)$, represent lth derivatives of $x(\alpha)$, $y(\alpha)$, $r(\alpha)$, $\pi(\alpha)$, $s(\alpha)$, $f(\alpha)$ and $g(\alpha)$ respectively, with respect to α. For simplicity we suppress α from the argument. Differentiating (1.3) j times at $\alpha = 1$ give

$$\begin{aligned} \textstyle\sum_{l=0}^{j} \begin{pmatrix} j \\ l \end{pmatrix} X^{(l)} r^{(j-l)} &= g^{(j)} X^{(0)} r^{(0)} + f^{(j)} \mu e \\ \textstyle\sum_{l=0}^{j} \begin{pmatrix} j \\ l \end{pmatrix} Y^{(l)} s^{(j-l)} &= g^{(j)} Y^{(0)} s^{(0)} + f^{(j)} \mu e \\ A x^{(j)} &= g^{(j)} \xi_b, \\ x^{(j)} + y^{(j)} &= g^{(j)} \xi_d \\ A^T \pi^{(j)} + r^{(j)} - s^{(j)} &= g^{(j)} \xi_c. \end{aligned} \tag{1.4}$$

Rearranging the terms in (1.4) yield:

$$\begin{bmatrix} X^{(0)} & 0 & R^{(0)} & 0 & 0 \\ 0 & Y^{(0)} & 0 & S^{(0)} & 0 \\ 0 & 0 & A & 0 & 0 \\ 0 & 0 & I & I & 0 \\ I & -I & 0 & 0 & A^T \end{bmatrix} \begin{pmatrix} r^{(j)} \\ s^{(j)} \\ x^{(j)} \\ y^{(j)} \\ \pi^{(j)} \end{pmatrix} = \begin{pmatrix} g^{(j)} X^{(0)} r^{(0)} + f^{(j)} \mu e - \sum_{l=1}^{j-1} \begin{pmatrix} j \\ l \end{pmatrix} X^{(l)} r^{(j-l)} \\ g^{(j)} Y^{(0)} s^{(0)} + f^{(j)} \mu e - \sum_{l=1}^{j-1} \begin{pmatrix} j \\ l \end{pmatrix} Y^{(l)} s^{(j-l)} \\ g^{(j)} \xi_b, \\ g^{(j)} \xi_d \\ g^{(j)} \xi_c. \end{pmatrix} \tag{1.5}$$

It is clear from (1.5) that all derivatives of $\Gamma(\alpha)$ can be computed recursively. The expressions in (1.5) can be simplified by defining $\Delta^{(l)} x = 1/l! x^{(l)}$, $\Delta^{(l)} y = 1/l! y^{(l)}$, $\Delta^{(l)} \pi = 1/l! \pi^{(l)}$, $\Delta^{(l)} r = 1/l! r^{(l)}$, $\Delta^{(l)} s = 1/l! s^{(l)}$; $u^{(l)} = 1/l![g^{(l)} X^{(0)} r^{(0)} + f^{(l)} \mu e] - \sum_{j=1}^{l-1} \Delta^{(l)} x \Delta^{(j-l)} r$, $v^{(l)} = 1/l![g^{(l)} Y^{(0)} s^{(0)} + f^{(l)} \mu e] - \sum_{j=1}^{l-1} \Delta^{(l)} Y \Delta^{(j-l)} s$, $\xi_b^{(l)} = 1/l! g^{(l)} \xi_b$, $\xi_d^{(l)} = 1/l! g^{(l)} \xi_d$, $\xi_c^{(l)} = 1/l! g^{(l)} \xi_c$ and $\Delta^{(l)} \Gamma = \Delta^{(l)} x, \Delta^{(l)} y, \Delta^{(l)} \pi, \Delta^{(l)} r, \Delta^{(l)} s$. After making these substitutions in (1.5), we have

$$M \Delta^{(j)} \Gamma = \begin{bmatrix} X^{(0)} & 0 & R^{(0)} & 0 & 0 \\ 0 & Y^{(0)} & 0 & S^{(0)} & 0 \\ 0 & 0 & A & 0 & 0 \\ 0 & 0 & I & I & 0 \\ I & -I & 0 & 0 & A^T \end{bmatrix} \begin{pmatrix} \Delta^{(j)} r \\ \Delta^{(j)} s \\ \Delta^{(j)} x \\ \Delta^{(j)} y \\ \Delta^{(j)} \pi \end{pmatrix} = \begin{pmatrix} u^{(j)} \\ v^{(j)} \\ \xi_b^{(j)} \\ \xi_d^{(j)} \\ \xi_c^{(j)} \end{pmatrix}. \tag{1.6}$$

Note from the form of (1.6) that computation of all the derivative involves only one factorization of M. The matrix M can be reduced by *a priori* defining some pivot sequence for its factorization. For example, if we substitute $\Delta^{(j)}r = {X^{(0)}}^{-1}(u^{(j)} - R^{(0)}\Delta^{(j)}x)$, $\Delta^{(j)}s = {Y^{(0)}}^{-1}(v^{(j)} - S^{(0)}\Delta^{(j)}y)$, obtained by eliminating the first two set of equations, into the last set of equations, we get

$$\begin{bmatrix} A & 0 & 0 \\ I & I & 0 \\ -{X^{(0)}}^{-1}R^{(0)} & {Y^{(0)}}^{-1}S^{(0)} & A^T \end{bmatrix} \begin{pmatrix} \Delta^{(j)}x \\ \Delta^{(j)}y \\ \Delta^{(j)}\pi \end{pmatrix} = \begin{pmatrix} \xi_b^{(j)} \\ \xi_d^{(j)} \\ \xi_c^{(j)} - {X^{(0)}}^{-1}u^{(j)} + {Y^{(0)}}^{-1}v^{(j)} \end{pmatrix}. \quad (1.7)$$

Now, if we use $\Delta^{(j)}y = \xi_d^{(j)} - \Delta^{(j)}x$ from the second set of equations in (1.7) in the third set we get

$$H \begin{pmatrix} \Delta^{(j)}\pi \\ \Delta^{(j)}x \end{pmatrix} \equiv \begin{bmatrix} -({X^{(0)}}^{-1}R^{(0)} + {Y^{(0)}}^{-1}S^{(0)}) & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} \Delta^{(j)}x \\ \Delta^{(j)}\pi \end{pmatrix} = \begin{pmatrix} \eta^{(j)} \\ \xi_b^{(j)} \end{pmatrix}, \quad (1.8)$$

where $\eta^{(j)} = \xi_c^{(j)} - {X^{(0)}}^{-1}u^{(j)} + {Y^{(0)}}^{-1}(v^{(j)} - S^{(0)}\xi_d^{(j)})$. In our implementation of the algorithms described in the next section we factor H after performing a simple symmetric scaling. H is a symmetric indefinite matrix and it can be factored symmetrically. The approach we used to compute sparse factors of H is described in Fourer and Mehrotra[6]. We refer the interested reader to that paper for details.

It is also possible to develop formulae for computing derivatives using normal equations approach. Let $\Theta = {X^{(0)}}^{-1}R^{(0)} + {Y^{(0)}}^{-1}S^{(0)}$. If second set of equations is used to eliminate $\Delta^{(j)}x$ from (1.8) we get

$$\begin{aligned}
\Delta^{(j)}\pi &= (A\Theta^{-1}A^T)^{-1}(\xi_b^{(j)} - A\Theta^{-1}({X^{(0)}}^{-1}u^{(j)} - {Y^{(0)}}^{-1}v^{(j)} + {Y^{(0)}}^{-1}S^{(0)}\xi_d^{(j)} - \xi_c^{(j)})), \\
\Delta^{(j)}x &= \Theta^{-1}({X^{(0)}}^{-1}u^{(j)} - {Y^{(0)}}^{-1}v^{(j)} + {Y^{(0)}}^{-1}S^{(0)}\xi_d^{(j)} - \xi_c^{(j)} + A^T\Delta^{(j)}\pi), \\
\Delta^{(j)}y &= \xi_d^{(j)} - \Delta^{(j)}x, \\
\Delta^{(j)}r &= {X^{(0)}}^{-1}u^{(j)} - {X^{(0)}}^{-1}R^{(0)}\Delta^{(j)}x, \\
\Delta^{(j)}s &= {Y^{(0)}}^{-1}v^{(j)} - {Y^{(0)}}^{-1}S^{(0)}\Delta^{(j)}y.
\end{aligned}$$

2. Basic Algorithms and their Implementation

In this section we describe two basic higher order algorithms (I, II). These algorithms are developed by specifying two different choices of $g(\alpha)$ and $f(\alpha)$. It is possible to take other choices of $g(\alpha)$ and $f(\alpha)$.

For Algorithms I we take

$$g(\alpha) = \alpha;\ f(\alpha) = \alpha(1-\alpha); \quad (2.1)$$

and for Algorithms II we take

$$g(\alpha) = \alpha;\ f(\alpha) = \alpha(1-\alpha)^2. \quad (2.2)$$

The choice of μ and the arrangement of computations for higher order methods resulting form using (2.1) and (2.2) is described later in this section.

At the beginning of iteration k we are given a point $x^k > 0,\ y^k > 0,\ \pi^k, r^k > 0,\ s^k > 0$. The approach we take to develop an implementation redefines a new trajectory at each iteration

by using (1.3). This allows for error in the numerical integration as there is no need to trace one trajectory accurately.

We now describe Algorithms I and II as we have implemented them.

At iteration k we let $\Gamma^{(0)} = (x^{(0)}, y^{(0)}, \pi^{(0)}, r^{(0)}, s^{(0)})=(x^k, y^k, \pi^k, r^k, s^k)$, and compute $\Delta^{(j)}\Gamma(1)$ for (2.1) and (2.2) using (1.8) for $j = 1, 2, \ldots lmax$ for a given $lmax$. The centering parameter μ needed for computing these derivative is generated adaptively. After computing the derivative we determine the order of polynomial to be used to generate a search direction. A heuristic is used to determine an order. Taylor polynomial is used to generate a search direction. We move in the search direction a certain distance (step factor) to the boundary to generate a new iterate. The step factor is determined adaptively. A detailed description of these steps is now given.

2.1 Centering

For both algorithms the centering parameter was estimated by using a heuristic proposed in Mehrotra[13]. The primal-dual affine scaling direction is used in estimating the centering parameter. The primal-dual affine scaling direction is defined as the first derivative of (1.3) for $g(\alpha) = \alpha$ and $\mu = 0$.

In Algorithms I the centering parameter appears in the computation of first derivative. In this case we first estimate μ, and then use it in the computation of first derivative of the trajectory. For Algorithms II the centering parameter appears for the first time while computing the second derivative. In this case the first derivative is same as the primal-dual affine scaling direction used in estimating the centering parameter. Hence, there is no need to recompute the first derivative. For a low order (particularly order 2) algorithm this is a significant advantage.

Let $p_x, p_y, p_\pi, p_r, p_s$ be the primal-dual affine scaling direction at the current point defined as above. Let

$$\begin{aligned}
\alpha_x &= \min\{x_i^k/(p_x)_i \mid (p_x)_i > 0\},\\
\alpha_y &= \min\{y_i^k/(p_y)_i \mid (p_y)_i > 0\},\\
\alpha_r &= \min\{r_i^k/(p_r)_i \mid (p_r)_i > 0\},\\
\alpha_s &= \min\{s_i^k/(p_s)_i \mid (p_s)_i > 0\},\\
\alpha_p &= \min(\alpha_x, \alpha_y, 1),\\
\alpha_d &= \min(\alpha_r, \alpha_s, 1).
\end{aligned}$$

The centering parameter is estimated as follows. We first compute

$$\mu = \left(\frac{(x^k - \alpha_p p_x)^T(r^k - \alpha_d p_r) + (y^k - \alpha_p p_y)^T(s^k - \alpha_d p_s)}{x^{k^T} r^k + y^{k^T} s^k}\right)^3 \frac{(x^{k^T} r^k + y^{k^T} s^k)}{2n}.$$

If

$$\frac{\|p_x\|^2 + \|p_y\|^2 + \|p_r\|^2 + \|p_s\|^2}{x^{k^T} r^k + y^{k^T} s^k} \geq 1.1,$$

then we reset

$$\mu = \mu/\min(\alpha_p, \alpha_d).$$

It is useful to note that the choice of $f(\alpha)$ from (2.1) and (2.2) for Algorithms I and II respectively result in a centering term as well as a "negative centering" term while computing derivatives. For Algorithms I centering term appears during the computations for first derivative, while the negative centering term appears during the computations for second derivative. For Algorithms II, centering appears in the second derivative, while negative centering in the third derivative.

2.2 Order of Taylor Polynomial

The derivative information is used to build a Taylor polynomial, which approximates the underlying trajectory. A Taylor polynomial is constructed as follows.

$$\Gamma(1-\alpha, l) \equiv \Gamma(1) + p_l(\Gamma, \alpha) \equiv \Gamma(1) + \sum_{j=1}^{l} (-\alpha)^j \Delta^{(j)}\Gamma(1). \tag{2.3}$$

Here $\Delta^{(j)}\Gamma(1)$ is computed from (1.6). $p_l(x,\alpha)$, $p_l(y,\alpha)$, $p_l(\pi,\alpha)$, $p_l(r,\alpha)$, $p_l(s,\alpha)$ are defined in a way similar to $p_l(\Gamma,\alpha)$.

Early in our experiments we found that the choice of the order of polynomial is important to the stability of the basic algorithm. For several problems a fixed choice of the order of polynomial at all iterations, while keeping all other aspects of the algorithm unchanged, failed in achieving desirable accuracy in the solution. The same problems were successfully solved if a lower order polynomial (of fixed order) was used. An examination of computational results revealed that these failures were not due to numerical errors. It appears that the failures are due to the "bad" search direction often computed by using a higher order polynomial.

The best way to deal with such situation is to use a progress function and perform a higher dimensional search. However, the choice of an appropriate progress function in the current situation is not completely understood. In our implementation we adaptively decided an appropriate order of polynomial by using a simple heuristic. On all of the experiments that we have performed so far, the performance of this heuristic has been satisfactory. We now describe our heuristic. Let

$$\begin{aligned}
\alpha_l(x) &= \max\{\alpha \mid x^k + p_l(x,\alpha) \geq 0 \text{ for all } 0 < \alpha \leq \alpha_l(x)\}, \quad l = 1, \ldots lmax, \\
\alpha_l(y) &= \max\{\alpha \mid y^k + p_l(y,\alpha) \geq 0 \text{ for all } 0 < \alpha \leq \alpha_l(y)\}, \quad l = 1, \ldots lmax, \\
\alpha_l(r) &= \max\{\alpha \mid r^k + p_l(r,\alpha) \geq 0 \text{ for all } 0 < \alpha < \alpha_l(r)\}, \quad l = 1, \ldots lmax, \\
\alpha_l(s) &= \max\{\alpha \mid s^k + p_l(s,\alpha) \geq 0 \text{ for all } 0 < \alpha < \alpha_l(s)\}, \quad l = 1, \ldots lmax,
\end{aligned}$$

$$\begin{aligned}
\alpha_l(p) &= \min(\alpha_l(x), \alpha_l(y), 1), \\
\alpha_l(d) &= \min(\alpha_l(r), \alpha_l(s), 1), \\
\alpha_l &= \min(\alpha_l(p), \alpha_l(d)).
\end{aligned}$$

Here $lmax$ is the maximum order of polynomial to be used. It is given *a priori* based on other considerations. $\alpha_l(p)$ is the maximum possible step along polynomial $p_l(x,\alpha)$ in the primal space which does not violate non-negativity. The restriction of maximum step size one ensures that we do not go beyond a feasible point. $\alpha_l(d)$ has a similar interpretation. α_l is the maximum possible step along polynomial $p_l(\Gamma,\alpha)$. Now we find

$$l^* = argmax\{\alpha_l\}$$

and take

$$l_p^* = \max(l^*, argmax\{\alpha_l(p), l = l^*, \ldots lmax\}),$$
$$l_d^* = \max(l^*, argmax\{\alpha_l(d), l = l^*, \ldots lmax\}).$$

to be the order of polynomials for primal and dual spaces. The use of different order of polynomials in primal and dual spaces generally results in slightly superior performance. This procedure was used for all the results reported in this paper.

2.3 Search Direction and Step Factor

For both algorithms after the order of Taylor polynomial is decided we compute the search directions in the primal and dual spaces as

$$d_x = -p_{l_p^*}(x, \alpha_{l_p^*}(p)), \quad d_y = -p_{l_p^*}(y, \alpha_{l_p^*}(p))$$
$$d_\pi = -p_{l_d^*}(\pi, \alpha_{l_d^*}(d)), \quad d_r = -p_{l_d^*}(r, \alpha_{l_d^*}(d)), \quad d_s = -p_{l_d^*}(s, \alpha_{l_d^*}(d)).$$

A step factor along directions d_x, d_y and d_π, d_r, d_s is computed as follows. This procedure is a straight forward generalization of a similar procedure in Mehrotra[13] for the bounded variable case. We first find

$$\begin{aligned} l_x &\equiv \text{argmin}\{x_i^k/(d_x)_i \mid (d_x)_i > 0,\ i = 1, 2, \ldots n\}, \\ l_y &\equiv \text{argmin}\{y_i^k/(d_y)_i \mid (d_y)_i > 0,\ i = 1, 2, \ldots n\}, \\ l_r &\equiv \text{argmin}\{r_i^k/(d_s)_i \mid (d_s)_i > 0,\ i = 1, 2, \ldots n\}, \\ l_s &\equiv \text{argmin}\{s_i^k/(d_s)_i \mid (d_s)_i > 0,\ i = 1, 2, \ldots n\}. \end{aligned}$$

If $\min\{x_i^k/(d_x)_i \mid (d_x)_i > 0,\ i = 1, 2, \ldots n\} \leq \min\{y_i^k/(d_y)_i \mid (d_y)_i > 0,\ i = 1, 2, \ldots n\}$, then we compute f_p such that

$$(x_{l_x}^k - f_p * (d_x)_{l_x})(r_{l_x}^k - f_p * (d_r)_{l_x}) = (x^k - d_x)^T(r^k - d_r) + (y^k - d_y)^T(s^k - d_s)/n\gamma_a,$$

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

otherwise

$$(y_{l_y}^k - f_p * (d_y)_{l_y})(s_{l_y}^k - f_p * (d_s)_{l_y}) = (x^k - d_x)^T(r^k - d_r) + (y^k - d_y)^T(s^k - d_s)/n\gamma_a,$$

$$f_p := \max(f_p, \gamma_f).$$

Similarly, for dual solutions if $\min\{r_i^k/(d_r)_i \mid (d_r)_i > 0,\ i = 1, 2, \ldots n\} \leq \min\{s_i^k/(d_s)_i \mid (d_s)_i > 0,\ i = 1, 2, \ldots n\}$, then we compute f_d such that

$$(r_{l_r}^k - f_d * (d_r)_{l_r})(x_{l_r}^k - f_d * (d_x)_{l_r}) = (x^k - d_x)^T(r^k - d_r) + (y^k - d_y)^T(s^k - d_s)/n\gamma_a,$$

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

otherwise

$$(s_{l_s}^k - f_d * (d_s)_{l_s})(y_{l_s}^k - f_d * (d_y)_{l_s}) = (x^k - d_x)^T(r^k - d_r) + (y^k - d_y)^T(s^k - d_s)/n\gamma_a,$$

$$f_d := \max(f_d, \gamma_f).$$

$\gamma_f = .9$ and $\gamma_a = 1/(1-\gamma_f)$ is used for all problems.

2.4 Solution Update

After computing the step factor and step direction for both algorithms we update the solutions from.

$$\begin{aligned} x^{k+1} &\leftarrow x^k - f_p d_x, \\ y^{k+1} &\leftarrow y^k - f_p d_y, \\ \pi^{k+1} &\leftarrow \pi^k - f_d d_\pi, \\ r^{k+1} &\leftarrow r^k - f_d d_r, \\ s^{k+1} &\leftarrow s^k - f_d d_s. \end{aligned}$$

2.5 Starting Point

A starting point was generated by using a generalization of the approach in Mehrotra[13] for the bounded variable case. As in that paper, to generate primal and dual starting points we consider the least-squares problems

$$\begin{aligned} \text{minimize} \quad & \|x, y\| \\ \text{s.t.} \quad & Ax = b, \\ & x + y = d, \end{aligned} \tag{2.4}$$

and

$$\begin{aligned} \text{minimize} \quad & \|r, s\| \\ \text{s.t.} \quad & A^T + r - s = c. \end{aligned} \tag{2.5}$$

Let $\tilde{x}$, $\tilde{y}$ and $\tilde{\pi}$, $\tilde{r}$, $\tilde{s}$ be the solutions for (2.4) and (2.5) respectively. These solutions can be obtained by factoring a matrix similar to H in (1.8). We take $\delta_p = \max(-1.5 * \min_i\{\tilde{x})_i\}, -1.5 * \min_i\{\tilde{y}_i\}, 0)$ and $\delta_d = \max(-1.5 * \min_i\{\tilde{r}_i\}, -1.5 * \min_i\{\tilde{s}_i\}, 0)$. Now

$$\tilde{\delta}_p = \delta_p + .5 * \frac{(\tilde{x} + \delta_p e)^T(\tilde{r} + \delta_d e) + (\tilde{y} + \delta_p e)^T(\tilde{s} + \delta_d e)}{\sum_{i=1}^n (\tilde{r}_i + \delta_d) + \sum_{i=1}^n (\tilde{s}_i + \delta_d)},$$

$$\tilde{\delta}_d = \delta_d + .5 * \frac{(\tilde{x} + \delta_p e)^T(\tilde{r} + \delta_d e) + (\tilde{y} + \delta_p e)^T(\tilde{s} + \delta_d e)}{\sum_{i=1}^n (\tilde{x}_i + \delta_p) + \sum_{i=1}^n (\tilde{y}_i + \delta_p)}.$$

Finally, the starting point is generated as $x^0 = \tilde{x} + \tilde{\delta}_p e$, $y^0 = \tilde{y} + \tilde{\delta}_p e$, $\pi^0 = \tilde{\pi}$, $r^0 = \tilde{r} + \tilde{\delta}_d e$, $s^0 = \tilde{s} + \tilde{\delta}_d e$.

3. Computational Experience

This section documents our computational experience with the higher order methods discussed in the earlier sections of this paper. The computational results reported here were obtained on 87 of the **netlib** problems. Problems *pilot87* and *dfl001* were too big for our current computing environment.

A FORTRAN 77 code implementing the algorithms described in the previous section was developed. All computations were performed on a SPARC-1 Sun workstation. The code was compiled

using FORTRAN compiler option "-O3."

Table 3.1 gives the data of the tested problems. The optimal objective value given in this table is from Bixby[3]. The objective value for problem *maros* is from the **netlib** index file. We have found [15] that the optimal values reported by Bixby are more accurate than those given in the current version of **netlib** index file. Problems with bounds and ranges are marked by B and R respectively. All problems were processed to remove easily identifiable null and fixed variables using the procedure CLEAN outlined in Adler *et al.*[2]. The results for *greenbea* and *greenbeb* were obtained after a small number of free variables, which had been modeled as differences of nonnegative variables, were identified explicitly to the code. No scaling was performed.

The matrix H in (1.8) was symmetrically factored at each iteration by using the procedure described in Fourer and Mehrotra[6]. Our detailed implementation also used the approach for handling free variables described in Mehrotra [14]. The work reported in [6, 14] is essential for the numerical stability of the results described here. Our earlier implementations of the same algorithms, which used normal equations approach to compute search direction and handling free variables by replacing it with two non-negative variables, were unsuccessful in satisfying the termination criterion (3.1) for several problems. In fact, the work reported in [6, 14] was motivated from the observed performance of the normal equations approach.

Tables 3.2 and 3.3 report computational results for the two basic algorithms. These tables report the number of iterations and the *cpu* time (in seconds) required to solve a problem using Algorithms I and II for $lmax = 1, 2, 3, \ldots 10$ and $lmax = 2, 3, \ldots 10$ respectively. In all cases the criterion

$$(b^T\pi - d^Ts - c^Tx)/(1 + |b^T\pi - d^Ts|) \leq 10^{-8} \tag{3.1}$$

was used to terminate an algorithm. This ensures about eight digits of accuracy in the objective value for all the problems. This same criterion was used in Lustig, Marsten and Shanno [12] to report their results. For all the problems at termination the solutions were acceptably accurate.

The results in Tables 3.2 and 3.3 show that all the problems were solved using the default setting to the desired precision. Significantly, the procedures for generating a starting point, computing centering parameter and step factor, which were used in Mehrotra [13] for the smaller problems (with no bounds) worked quiet satisfactorily for the entire test set for all the different higher order methods.

Now we discuss the performance of Algorithms I and Algorithms II with respect to the order of polynomial. For convenience we use Algorithm I.o to represent Algorithm I using $lmax$ =o. A similar convention would be used for Algorithms II.

Both algorithms show a significant decrease in the number of iterations as the order of polynomial is increased from order two. From the results in Mehrotra[13] and Lustig, Marsten and Shanno[12] we already know that a method using second order information saves 30 to 40% iterations over the first order method. For example, Algorithm I.2 saves 28% iterations over Algorithm I.1. The results here show that Algorithm I.3 and I.4 would save an additional 18%, and 25% iterations over I.2. If we go up to I.10, the savings in the number of iterations would be approximately 35%. For Algorithms II the respective savings in the number of iterations are 8%, 15% and 26%.

The results also show that the number of iterations reduce very slowly (if at all) beyond a certain order for both methods. This indicate that the improvement in the number of iterations

from methods based on using Taylor polynomials may be limited.

The savings in the number of iterations does not always translate into savings in computational time. This is because additional work is required to compute and use higher derivatives. For Algorithms I, I.3 saves about 11% time over I.2 on the average. This is still more than the time required by Algorithm II.2. Note that Algorithms I solve (1.8) because of the way the centering parameter is estimated. On the average Algorithm II.2 out performs all Algorithms II. However, as expected, on several problems for which the work involved in matrix factorization dominates the computational efforts of an iteration, II.3 or II.4 performs better. Most significant savings are for problems *bnl2, d2q06c, pilot,* which are the three most time consuming tested **netlib** problems. The savings from II.4 over II.2 for these problems are 12%, 10% and 16% respectively.

Name	Rows	Cols	Nonzeros	BR	Objective
25fv47	822	1571	11127		5.5018458883e+3
80bau3b	2263	9799	29063	B	9.8722419241e+5
adlittle	57	97	465		2.2549496316e+5
afiro	28	32	88		-4.6475314286e+2
agg	489	163	2541		-3.5991767287e+7
agg2	517	302	4515		-2.0239252356e+7
agg3	517	302	4531		1.0312115935e+7
bandm	306	472	2659		-1.5862801845e+2
beaconfd	174	262	3476		3.3592485807e+4
blend	75	83	521		-3.0812149846e+1
bnl1	644	1175	6129		1.9776295615e+3
bnl2	2325	3489	16124		1.8112365404e+3
boeing1	351	384	3865	BR	-3.3521356751e+2
boeing2	167	143	1339	BR	-3.1501872802e+2
bore3d	234	315	1525	B	1.3730803942e+3
brandy	221	249	2150		1.5185098965e+3
capri	272	353	1786	B	2.6900129138e+3
cycle	1904	2857	21322	B	-5.2263930249e+0
czprob	930	3523	14173	B	2.1851966989e+6
d2q06c	2172	5167	35674		1.2278421081e+5
degen2	445	534	4449		-1.4351780000e+3
degen3	1504	1818	26230		-9.8729400000e+2
e226	224	282	2767		-1.8751929066e+1
etamacro	401	688	2489	B	-7.5571523337e+2
fffff800	525	854	6235		5.5567956482e+5
finnis	498	614	2714	B	1.7279106560e+5
fit1d	25	1026	14430	B	-9.1463780924e+3
fit1p	628	1677	10894	B	9.1463780924e+3
fit2d	26	10500	138018	B	-6.8464293294e+4
fit2p	3001	13525	60784	B	6.8464293294e+4
forplan	162	421	4916	BR	-6.6421896127e+2
ganges	1310	1681	7021	B	-1.0958573613e+2
gfrd-pnc	617	1092	3467	B	6.9022359995e+6
greenbea	2393	5405	31499	B	-7.2555248130e+6
greenbeb	2393	5405	31499	B	-4.3022602612e+6
grow15	301	645	5665	B	-1.0687094129e+8
grow22	441	946	8318	B	-1.6083433648e+8
grow7	141	301	2633	B	-4.7787811815e+7
israel	175	142	2358		-8.9664482186e+5
kb2	44	41	291	B	-1.7499001299e+3
lotfi	154	308	1086		-2.5264706062e+1
maros	847	1443	10006	B	-5.8063743701e+4
nesm	663	2923	13988	BR	1.4076036488e+7

Table 3.1: Problem Data Summary (A-N)

Name	Rows	Cols	Nonzeros	BR	Objective
perold	626	1376	6026	B	-9.3807552782e+3
pilot.ja	941	1988	14706	B	-6.1131364656e+3
pilot.we	723	2789	9218	B	-2.7201075328e+6
pilot4	411	1000	5145	B	-2.5811392589e+3
pilotnov	976	2172	13129	B	-4.4972761882e+3
recipe	92	180	752	B	-2.6661600000e+2
sc105	106	103	281		-5.2202061212e+1
sc205	206	203	552		-5.2202061212e+1
sc50a	51	48	131		-6.4575077059e+1
sc50b	51	48	119		-7.0000000000e+1
scagr25	472	500	2029		-1.4753433061e+7
scagr7	130	140	553		-2.3313898243e+7
scfxm1	331	457	2612		1.8416759028e+4
scfxm2	661	914	5229		3.6660261565e+4
scfxm3	991	1371	7846		5.4901254550e+4
scorpion	389	358	1708		1.8781248227e+3
scrs8	491	1169	4029		9.0429695380e+2
scsd1	78	760	3148		8.6666666743e+1
scsd6	148	1350	5666		5.0500000078e+1
scsd8	398	2750	11334		9.0499999993e+2
sctap1	301	480	2052		1.4122500000e+3
sctap2	1091	1880	8124		1.7248071429e+3
sctap3	1481	2480	10734		1.4240000000e+3
seba	516	1028	4874	BR	1.5711600000e+4
share1b	118	225	1182		-7.6589318579e+4
share2b	97	79	730		-4.1573224074e+2
shell	537	1775	4900	B	1.2088253460e+9
ship04l	403	2118	8450		1.7933245380e+7
ship04s	403	1458	5810		1.7987147004e+6
ship08l	779	4283	17085		1.9090552114e+6
ship08s	779	2387	9501		1.9200982105e+6
ship12l	1152	5427	21597		1.4701879193e+6
ship12s	1152	2763	10941		1.4892361344e+6
sierra	1228	2036	9252	B	1.5394362184e+7
stair	357	467	3857	B	-2.5126695119e+2
standata	360	1075	3038	B	1.2576995000e+3
standmps	468	1075	3686	B	1.4060175000e+3
stocfor1	118	111	474		-4.1131976219e+4
stocfor2	2158	2031	9492		-3.9024408538e+4
tuff	334	587	4523	B	2.9214776509e-1
vtp.base	199	203	914	B	1.2983146246e+5
wood1p	245	2594	70216		1.4429024116e+0
woodw	1099	8405	37478		1.3044763331e+0

Table 3.1: Problem Data Summary (P-W)

	Order of Polynomial									
Name	1	2	3	4	5	6	7	8	9	10
25fv47	42	31	24	21	20	19	20	18	17	18
25fv47	190.23	159.67	131.55	125.55	124.74	126.04	140.49	139.54	133.62	149.80
80bau3b	81	60	47	39	43	35	36	42	40	42
80bau3b	516.35	446.56	528.75	482.60	606.77	519.68	631.13	816.88	859.68	970.33
adlittle	17	12	10	9	8	8	7	8	8	7
adlittle	0.84	0.85	0.90	0.89	0.97	1.07	1.09	1.39	1.63	1.54
afiro	11	8	6	6	6	5	5	5	5	5
afiro	0.18	0.21	0.18	0.25	0.29	0.27	0.31	0.36	0.42	0.46
agg	32	24	20	20	20	20	20	18	20	19
agg	8.66	8.44	7.86	8.97	9.64	11.54	13.01	12.45	15.83	16.20
agg2	31	21	21	17	17	17	17	16	16	16
agg2	61.06	46.27	49.42	43.45	46.01	48.45	50.81	50.96	53.96	55.90
agg3	29	22	20	19	16	16	17	16	17	16
agg3	56.35	46.89	46.38	46.68	43.32	45.45	50.11	50.05	55.56	55.00
bandm	26	18	15	13	13	13	11	11	10	10
bandm	8.66	6.26	6.20	6.28	6.93	7.69	7.45	8.19	7.55	8.29
beaconfd	12	7	7	6	5	5	5	5	5	4
beaconfd	2.72	1.96	3.45	3.27	3.10	3.30	3.54	3.80	4.07	3.93
blend	16	10	9	8	7	8	8	7	7	6
blend	1.16	0.82	1.04	1.07	0.95	1.36	1.51	1.33	1.64	1.43
bnl1	45	32	24	26	20	25	21	20	19	19
bnl1	50.87	42.20	37.29	45.02	39.83	54.28	51.06	53.21	55.51	60.74
bnl2	60	44	34	30	26	26	26	23	24	23
bnl2	1021.16	679.38	559.09	518.96	479.84	501.90	574.07	482.60	524.70	556.38
boeing1	38	29	21	20	18	17	16	16	16	15
boeing1	22.88	20.04	16.53	18.19	19.78	19.80	20.86	24.43	25.94	26.68
boeing2	28	20	16	16	14	14	13	12	13	14
boeing2	5.19	4.43	4.08	5.09	5.49	5.94	5.98	6.20	7.49	8.67
bore3d	25	19	16	17	13	12	13	13	13	12
bore3d	2.72	2.46	2.43	2.97	2.65	2.81	3.53	3.96	4.51	4.46
brandy	27	23	18	17	16	19	17	17	16	17
brandy	8.61	8.05	7.34	7.70	7.84	10.09	9.91	10.79	11.43	12.15
capri	47	22	17	17	15	14	12	13	14	12
capri	12.13	8.60	8.81	10.11	9.61	10.12	10.10	12.44	14.09	13.49
cycle	47	36	27	25	23	23	22	26	24	22
cycle	237.93	203.28	176.64	171.50	173.18	196.93	183.57	229.98	224.78	207.09
czprob	53	42	33	31	30	27	27	27	24	27
czprob	56.42	52.21	51.29	57.24	65.42	68.35	77.92	87.72	87.52	109.30
d2q06c	50	38	29	25	22	23	22	21	21	21
d2q06c	1482.07	1096.73	881.93	815.22	756.07	812.47	794.77	828.23	824.82	887.76
degen2	19	12	10	10	9	8	9	8	9	8
degen2	50.57	25.29	30.49	28.43	31.45	30.37	32.69	35.61	41.69	31.50
degen3	27	18	15	13	13	12	11	11	10	11
degen3	854.52	638.92	565.64	488.10	424.16	584.26	434.98	407.33	434.20	544.45
e226	30	23	18	16	14	15	14	13	14	13
e226	12.21	10.71	9.65	9.79	9.64	11.30	11.63	11.86	13.76	13.87
etamacro	44	31	26	24	21	20	21	21	20	20
etamacro	36.21	29.35	28.71	27.95	27.21	29.84	33.88	35.13	36.20	39.38
fffff800	47	35	30	28	31	26	27	26	26	24
fffff800	52.98	41.61	45.81	42.10	57.13	47.21	53.23	60.38	65.07	61.33
finnis	43	35	26	20	19	18	16	16	19	19
finnis	19.00	18.74	17.40	16.04	17.73	19.29	19.68	21.83	28.47	31.88
fit1d	28	19	17	17	16	16	16	16	17	16
fit1d	25.29	20.26	21.55	24.98	26.92	30.47	34.29	38.80	45.17	47.37
fit1p	26	19	17	17	14	15	16	15	16	17
fit1p	22.00	19.31	21.13	24.75	23.69	28.78	34.19	36.33	43.22	51.55
fit2d	37	28	22	19	18	21	17	16	15	14
fit2d	330.23	290.13	274.34	278.82	301.14	397.48	366.23	389.99	409.37	428.66
fit2p	33	22	18	18	15	13	13	13	13	13
fit2p	196.22	161.23	169.01	203.38	202.55	207.33	236.70	273.54	308.36	345.03

Table 3.2: Performance of Algorithms I (A-Fi).

	Order of Polynomial									
Name	1	2	3	4	5	6	7	8	9	10
forplan1	44	29	27	20	21	20	19	18	17	18
forplan1	18.29	12.63	14.62	11.68	13.82	14.81	15.65	16.23	16.92	19.40
ganges	30	19	17	15	15	14	13	13	12	12
ganges	49.36	34.92	34.73	31.39	68.85	40.35	38.35	42.44	43.46	50.87
gfrd-pnc	28	18	16	13	13	12	12	12	12	11
gfrd-pnc	12.22	9.65	11.11	10.49	13.03	13.47	15.49	18.19	19.95	20.70
greenbea	56	49	45	39	31	40	43	29	37	36
greenbea	261.21	256.26	266.36	269.52	235.54	319.86	372.69	286.53	375.33	405.09
greenbeb	67	51	39	34	35	33	29	30	29	29
greenbeb	276.94	240.93	212.19	208.93	237.17	247.23	244.23	272.12	287.65	311.03
grow15	19	14	11	10	9	9	9	10	9	8
grow15	15.12	12.86	12.16	12.60	12.71	14.34	15.94	19.35	19.23	18.86
grow22	19	14	12	11	11	10	10	10	10	11
grow22	21.22	18.30	18.59	19.41	21.74	22.41	24.85	27.33	30.15	35.84
grow7	17	12	11	9	9	10	9	8	8	8
grow7	6.13	5.13	5.45	5.30	5.94	7.37	7.34	7.23	8.03	8.81
israel	38	28	23	21	19	18	18	18	18	17
israel	9.32	8.43	8.19	8.54	8.00	8.48	9.93	10.28	11.32	12.51
kb2	23	19	16	16	15	15	14	14	14	14
kb2	0.94	0.85	0.93	1.18	1.24	1.48	1.54	1.74	1.95	2.18
lotfi	22	15	12	11	10	10	10	10	9	9
lotfi	3.16	2.67	2.72	2.98	3.20	3.67	4.13	4.95	4.79	5.32
maros	39	28	22	21	19	22	19	20	17	18
maros	72.91	59.71	53.83	55.93	55.84	68.52	65.76	73.45	70.67	79.51
nesm	57	43	32	29	28	26	24	24	25	25
nesm	136.17	118.25	104.65	108.93	119.78	124.48	130.01	143.98	164.42	179.37
perold	48	36	28	28	27	24	24	24	26	22
perold	150.55	124.65	102.55	109.41	121.29	107.05	112.93	121.26	155.26	124.08
pilot	71	58	44	44	41	35	36	35	38	40
pilot	3280.71	2786.92	2190.76	2256.06	2167.92	1903.00	1998.60	2104.75	2224.98	2374.10
pilot4	56	43	35	34	31	40	32	27	31	34
pilot4	58.58	50.66	48.14	52.08	52.32	71.77	64.61	60.25	74.92	89.38
pilot.ja	53	46	40	36	34	33	34	35	35	33
pilot.ja	392.13	348.70	331.89	304.40	302.34	306.44	328.72	353.76	370.07	363.25
pilotnov	35	27	22	19	19	19	16	17	19	17
pilotnov	232.94	202.17	178.31	153.45	161.58	182.25	152.76	168.78	208.00	196.76
pilot.we	81	59	46	42	39	38	37	38	34	39
pilot.we	147.01	123.58	114.01	119.18	125.60	136.80	153.36	169.26	170.66	209.26
recipe	14	10	8	8	7	7	7	7	6	6
recipe	1.32	1.16	1.18	1.36	1.40	1.56	1.80	2.04	1.97	2.17
sc105	15	9	8	7	6	6	5	5	5	5
sc105	1.01	0.86	0.93	0.97	0.89	1.13	1.00	1.12	1.22	1.40
sc205	17	12	10	9	10	7	7	6	7	7
sc205	2.07	1.83	1.95	2.36	2.74	2.30	2.79	2.57	3.26	3.83
sc50a	12	8	7	6	6	5	5	5	5	5
sc50a	0.37	0.31	0.40	0.36	0.42	0.42	0.47	0.52	0.59	0.65
sc50b	10	7	6	5	5	5	4	4	4	4
sc50b	0.35	0.31	0.33	0.30	0.39	0.45	0.38	0.42	0.47	0.51
scagr25	26	17	14	13	12	12	12	11	10	10
scagr25	6.43	5.20	5.58	6.27	6.77	7.79	9.53	9.27	9.62	10.61
scagr7	19	14	12	10	11	10	10	9	10	9
scagr7	1.27	1.17	1.30	1.31	1.67	1.78	2.00	2.08	2.51	2.60
scfxm1	28	19	16	16	13	13	12	13	12	12
scfxm1	11.15	8.97	8.97	10.17	9.47	10.59	10.97	13.04	13.41	14.48
scfxm2	45	22	18	17	15	13	13	13	13	12
scfxm2	48.40	22.51	20.07	23.38	21.59	21.20	25.25	27.77	28.53	28.99
scfxm3	34	22	19	17	16	14	16	12	13	15
scfxm3	43.42	30.70	34.19	35.39	37.17	36.73	45.43	36.19	42.90	53.38
scorpion	18	12	10	9	9	8	8	8	8	8
scorpion	5.69	5.01	4.15	4.38	5.72	4.99	5.49	6.07	7.46	8.04

Table 3.2: Performance of Algorithms I (Fo-Sco).

	Order of Polynomial									
Name	1	2	3	4	5	6	7	8	9	10
scrs8	33	23	17	17	14	14	14	14	14	13
scrs8	16.85	14.53	13.79	16.19	14.85	16.88	19.20	22.76	25.22	26.12
scsd1	12	8	7	7	7	6	6	5	6	6
scsd1	3.39	2.95	3.32	3.72	4.31	4.38	4.95	4.74	6.21	6.95
scsd6	16	11	9	9	9	7	8	8	7	7
scsd6	8.22	7.00	7.21	8.46	9.76	9.07	11.43	13.02	12.88	14.36
scsd8	15	10	9	8	7	7	7	6	6	6
scsd8	16.38	13.53	15.02	15.97	16.44	18.98	21.22	20.94	23.17	25.73
sctap1	22	16	13	13	12	12	11	12	11	11
sctap1	5.84	5.23	5.39	6.39	6.88	7.94	8.43	10.29	10.62	12.23
sctap2	21	15	12	10	10	9	9	9	9	8
sctap2	34.34	27.09	26.12	25.63	28.30	28.80	32.20	34.90	38.73	38.41
sctap3	22	17	13	12	11	10	10	9	9	9
sctap3	44.28	37.75	35.11	37.00	42.07	40.50	45.09	45.40	50.55	55.18
seba	27	20	17	15	13	13	12	12	13	12
seba	12.23	10.29	11.67	12.51	12.13	14.76	14.82	17.49	20.45	21.21
share1b	36	24	20	19	17	18	17	18	15	18
share1b	4.52	3.42	3.72	4.10	4.06	5.04	5.20	6.34	5.86	7.77
share2b	18	12	10	9	9	9	9	8	8	8
share2b	1.69	1.45	1.46	1.38	1.57	1.81	2.15	1.99	2.20	2.43
shell	33	23	19	18	16	15	15	15	14	13
shell	20.28	16.75	17.27	19.68	20.40	22.38	25.03	28.49	29.76	30.88
ship04l	19	13	11	10	9	9	8	8	8	7
ship04l	13.89	12.86	12.63	13.75	14.47	17.66	16.96	19.17	22.32	20.94
ship04s	20	14	11	11	10	9	10	10	8	8
ship04s	9.86	8.50	8.58	10.83	10.76	11.28	14.11	16.01	14.70	16.29
ship08l	22	15	13	12	11	11	10	11	11	11
ship08l	28.16	22.31	25.99	28.55	28.81	33.04	34.23	42.27	47.13	52.43
ship08s	22	14	12	11	10	10	11	9	10	9
ship08s	14.93	11.19	12.85	13.32	14.26	17.07	21.02	18.72	23.48	23.28
ship12l	25	19	14	14	13	13	12	12	12	12
ship12l	42.42	39.71	35.28	44.60	48.24	54.72	55.34	62.61	69.92	80.25
ship12s	25	18	13	14	13	12	12	12	13	12
ship12s	20.62	18.46	15.98	21.18	23.10	23.32	27.75	31.08	35.91	38.22
sierra	32	23	19	16	15	15	13	13	14	14
sierra	79.99	71.28	83.52	67.41	69.61	68.98	74.15	74.05	86.58	93.15
stair	25	16	14	14	11	11	11	10	11	11
stair	22.67	14.77	16.84	15.70	13.85	14.86	18.39	15.86	18.37	19.60
standata	20	13	12	12	12	12	10	9	10	11
standata	9.65	7.26	8.28	9.89	11.69	13.98	12.97	14.06	16.48	20.25
standmps	24	17	15	14	13	14	12	14	15	14
standmps	14.20	11.47	12.54	13.67	14.95	18.96	18.11	23.09	27.45	29.44
stocfor1	21	15	15	13	12	12	11	10	10	11
stocfor1	1.42	1.30	1.70	1.65	1.78	2.01	2.29	2.24	2.44	2.96
stocfor2	36	26	20	19	17	16	16	15	16	16
stocfor2	69.10	56.80	53.75	58.20	59.96	60.67	67.32	70.49	86.40	91.66
tuff	30	21	19	19	15	17	15	16	14	15
tuff	20.56	16.55	17.08	18.91	17.03	20.36	19.98	22.99	22.26	25.39
vtp.base	22	17	17	18	15	15	15	14	14	14
vtp.base	1.62	1.42	1.86	2.43	2.43	2.86	3.24	3.51	3.91	4.58
wood1p	22	21	20	19	18	22	19	20	16	19
wood1p	106.68	109.42	101.22	103.17	116.08	132.57	123.68	137.39	118.55	146.37
woodw	38	31	25	25	21	23	22	24	21	21
woodw	178.64	173.83	150.15	158.59	149.99	186.52	197.82	232.43	223.67	243.16
Total	2737	1992	1635	1511	1398	1379	1327	1297	1296	1290
cpu(sec)	11414	9124	8194	8076	8098	8266	8648	9130	9655	10527

Table 3.2: Performance of Algorithms I (Scr-Z).

	Order of Polynomial								
Name	2	3	4	5	6	7	8	9	10
25fv47	26	25	21	20	22	20	20	22	18
25fv47	125.99	134.54	116.93	118.39	137.74	135.11	142.06	169.11	143.29
80bau3b	45	41	39	40	32	31	31	31	33
80bau3b	318.23	365.21	425.92	445.17	415.74	449.21	498.91	559.73	669.61
adlittle	11	10	9	8	8	8	8	7	8
adlittle	0.68	0.74	0.90	0.85	1.01	1.20	1.39	1.30	1.73
afiro	7	6	6	6	6	5	5	5	5
afiro	0.13	0.16	0.21	0.23	0.29	0.29	0.33	0.38	0.45
agg	23	24	24	23	22	21	23	22	17
agg	6.53	8.16	9.50	10.61	11.58	12.73	15.41	16.47	14.66
agg2	22	22	20	20	16	17	18	16	15
agg2	45.46	48.95	47.21	49.96	40.69	48.64	54.50	51.89	51.18
agg3	21	21	19	27	17	17	18	16	16
agg3	42.93	46.15	45.17	64.32	45.38	47.94	53.30	51.57	51.09
bandm	17	16	14	12	12	12	11	11	11
bandm	5.35	5.86	6.06	5.47	6.11	7.61	7.80	8.52	9.16
beaconfd	7	7	6	5	5	5	5	5	4
beaconfd	1.83	3.19	3.10	2.94	3.12	3.43	3.63	3.92	3.82
blend	10	10	8	8	7	7	8	7	7
blend	0.71	1.00	0.85	1.12	1.01	1.15	1.57	1.55	1.73
bnl1	30	28	25	23	22	22	21	23	25
bnl1	35.58	38.70	40.39	41.77	45.34	50.38	52.75	63.31	74.99
bnl2	36	32	28	29	27	24	26	25	24
bnl2	541.07	510.18	474.87	508.24	497.35	469.84	547.53	529.78	532.83
boeing1	25	22	21	19	19	18	19	17	17
boeing1	14.71	15.71	17.30	17.27	20.77	22.14	25.65	24.87	27.54
boeing2	19	17	15	14	14	14	14	13	12
boeing2	3.81	3.73	4.38	4.35	4.99	5.67	6.94	6.68	6.88
bore3d	17	16	16	17	16	17	17	17	18
bore3d	1.67	1.99	2.65	3.02	3.48	4.20	4.66	5.03	6.28
brandy	18	15	17	13	13	16	16	13	12
brandy	5.89	5.65	6.63	5.39	6.43	8.24	8.51	8.69	7.69
capri	20	18	17	14	15	14	14	14	12
capri	6.74	7.37	8.87	7.97	9.68	10.97	12.12	12.97	12.24
cycle	30	26	23	24	21	21	22	21	22
cycle	151.52	154.52	142.26	160.68	165.87	152.25	192.97	177.12	217.70
czprob	36	33	31	29	28	26	29	29	28
czprob	37.10	43.73	50.37	57.16	63.53	68.74	86.70	98.40	106.67
d2q06c	30	28	24	25	24	23	24	22	24
d2q06c	855.79	846.01	770.03	822.93	815.84	813.21	917.84	836.43	941.48
degen2	12	11	11	9	8	8	9	7	8
degen2	27.47	23.99	32.50	26.92	26.22	33.52	34.37	28.55	31.23
degen3	16	15	15	14	12	14	13	12	12
degen3	441.55	393.08	557.52	473.81	520.61	463.82	498.11	396.26	477.56
e226	20	19	16	16	15	14	14	13	13
e226	8.55	9.22	9.15	10.12	10.56	10.89	11.48	12.36	13.27
etamacro	30	28	23	22	21	22	21	20	22
etamacro	25.83	26.74	24.94	26.50	27.95	31.79	33.41	34.63	41.23
fffff800	36	35	31	34	32	31	30	34	33
fffff800	38.65	47.33	42.69	55.93	53.87	57.51	65.08	79.14	79.30
finnis	25	24	19	20	18	17	17	16	16
finnis	11.89	14.06	13.72	16.85	17.75	19.33	21.55	23.03	25.87
fit1d	18	20	18	22	16	18	19	18	18
fit1d	16.86	22.35	23.94	33.13	28.32	35.83	42.99	45.54	50.44
fit1p	18	18	17	21	16	17	18	16	17
fit1p	15.60	19.30	21.77	31.60	28.04	34.11	40.89	41.07	48.66
fit2d	23	25	19	20	18	19	18	16	17
fit2d	209.56	274.96	249.80	306.22	322.01	384.29	405.38	410.86	484.16
fit2p	20	19	17	15	15	18	16	16	14
fit2p	122.43	152.88	172.79	183.94	214.90	296.87	306.44	348.82	351.30

Table 3.3: Performance of Algorithms II (A-Fi)

	Order of Polynomial								
Name	2	3	4	5	6	7	8	9	10
forplan1	23	24	21	21	18	17	20	19	16
forplan1	9.05	11.10	11.31	12.84	13.43	13.17	17.12	18.86	16.71
ganges	18	17	16	14	16	14	13	16	13
ganges	26.83	31.24	67.18	30.71	49.99	38.16	39.95	60.15	48.31
gfrd-pnc	16	15	14	13	12	12	12	13	12
gfrd-pnc	7.27	9.01	9.84	11.12	12.33	14.85	17.02	20.29	21.19
greenbea	40	38	32	30	31	32	31	31	28
greenbea	194.03	209.03	202.18	203.77	239.15	270.22	279.35	302.55	308.41
greenbeb	39	35	34	33	31	34	32	32	31
greenbeb	167.73	173.67	190.89	207.13	217.22	263.64	272.06	296.57	313.69
grow15	11	11	10	10	9	10	9	9	9
grow15	9.47	11.00	11.64	13.00	13.40	16.40	16.50	18.31	20.19
grow22	13	11	11	10	10	10	9	9	9
grow22	15.26	15.48	17.90	18.49	20.84	23.35	23.44	25.93	28.63
grow7	11	11	9	10	9	8	8	8	8
grow7	4.21	4.94	4.84	5.95	6.26	6.28	6.85	7.66	8.43
israel	24	23	21	19	20	19	18	18	19
israel	6.66	6.85	7.75	7.46	8.72	9.87	10.32	11.31	12.96
kb2	19	18	16	15	14	15	14	14	14
kb2	0.79	0.93	1.00	1.12	1.21	1.53	1.62	1.83	2.05
lotfi	14	14	11	11	12	10	10	10	10
lotfi	2.14	2.68	2.67	3.15	4.06	3.84	4.64	5.27	5.47
maros	26	26	21	20	19	20	19	19	19
maros	50.60	59.38	51.93	54.23	56.65	63.00	66.78	72.39	77.85
nesm	32	30	31	29	26	25	26	26	27
nesm	79.64	87.89	105.27	112.61	112.75	125.97	145.41	160.49	183.55
perold	32	30	27	27	26	25	24	25	24
perold	106.49	102.69	100.60	111.64	110.04	118.24	116.80	126.38	129.39
pilot	40	38	32	34	36	31	33	36	35
pilot	1933.19	1902.25	1664.10	1800.73	2007.01	1730.06	1975.13	2070.78	2070.80
pilot4	44	39	39	40	35	34	35	35	36
pilot4	46.50	47.65	54.54	61.23	60.31	63.57	73.06	79.09	89.59
pilot.ja	46	47	43	41	42	40	39	37	36
pilot.ja	336.45	359.04	346.30	349.62	370.79	370.83	377.35	379.21	383.15
pilotnov	24	21	21	19	18	18	16	16	17
pilotnov	175.08	153.22	172.85	155.53	167.88	163.81	154.22	174.64	191.91
pilot.we	49	44	38	37	35	33	34	36	36
pilot.we	91.18	97.39	98.34	109.71	117.72	129.23	143.02	169.62	186.27
recipe	10	9	7	7	8	7	7	7	7
recipe	1.09	1.21	1.21	1.43	1.66	1.76	1.97	2.34	2.45
sc105	9	8	7	6	6	5	5	5	5
sc105	0.64	0.76	0.78	0.84	0.99	0.94	1.08	1.18	1.31
sc205	11	10	8	8	9	7	6	7	6
sc205	1.39	1.71	1.72	2.02	2.82	2.40	2.41	3.04	3.01
sc50a	8	7	7	6	5	5	5	5	5
sc50a	0.29	0.30	0.41	0.39	0.39	0.45	0.51	0.56	0.63
sc50b	7	6	5	5	4	4	4	4	4
sc50b	0.28	0.29	0.26	0.36	0.30	0.35	0.40	0.43	0.49
scagr25	17	15	13	12	12	13	11	11	11
scagr25	4.36	5.07	5.51	6.17	7.63	8.95	8.71	9.98	11.02
scagr7	13	12	11	10	9	9	9	9	11
scagr7	0.89	1.11	1.28	1.39	1.52	1.69	1.94	2.14	3.00
scfxm1	18	17	15	14	13	12	12	12	13
scfxm1	7.53	9.38	8.75	9.27	9.91	10.21	11.49	12.69	15.00
scfxm2	20	19	16	16	15	15	15	14	12
scfxm2	16.46	20.64	18.38	23.03	22.34	26.67	29.85	30.64	27.73
scfxm3	20	19	17	15	15	14	13	14	12
scfxm3	24.39	30.74	29.11	29.73	33.46	35.10	36.55	43.40	41.49
scorpion	12	10	9	9	9	8	8	9	8
scorpion	3.66	3.70	4.04	4.48	5.06	5.14	5.72	7.50	6.91

Table 3.3: Performance of Algorithms II (Fo-Sco)

	Order of Polynomial								
Name	2	3	4	5	6	7	8	9	10
scrs8	21	18	17	16	18	16	15	14	14
scrs8	10.45	11.50	13.44	15.06	20.42	19.97	22.57	22.73	25.54
scsd1	8	7	7	7	6	6	6	6	6
scsd1	2.53	2.96	3.37	3.90	4.04	4.63	5.26	5.86	6.58
scsd6	10	9	9	8	8	7	7	7	7
scsd6	5.53	6.31	7.65	8.04	9.32	9.47	10.68	12.17	13.62
scsd8	10	9	8	7	7	7	7	6	6
scsd8	11.51	13.12	14.09	14.96	17.43	19.78	22.52	21.89	24.36
sctap1	15	13	13	13	12	12	11	11	11
sctap1	4.50	4.55	5.68	7.12	7.33	8.47	8.88	10.00	11.23
sctap2	13	12	10	9	9	9	10	9	9
sctap2	21.07	25.68	23.29	23.82	26.80	30.20	38.56	36.71	40.39
sctap3	14	13	11	12	10	9	9	9	9
sctap3	27.75	31.04	31.08	38.58	37.24	37.98	42.48	47.66	52.34
seba	18	16	14	15	13	13	12	12	12
seba	7.90	9.04	9.94	12.50	12.86	14.95	15.75	18.48	20.07
share1b	22	21	19	17	16	15	17	17	17
share1b	2.88	3.45	3.46	3.92	4.28	4.32	5.47	6.18	6.83
share2b	12	11	10	10	9	8	9	9	9
share2b	1.08	1.42	1.54	1.60	1.88	1.66	2.23	2.51	2.54
shell	20	19	17	16	16	14	15	13	13
shell	12.36	15.06	16.41	19.29	21.57	22.86	27.42	26.49	29.21
ship04l	12	11	10	9	9	8	8	7	7
ship04l	9.25	11.86	12.12	13.11	16.14	15.66	18.05	17.82	19.86
ship04s	13	12	11	11	10	10	11	9	10
ship04s	6.75	8.10	8.92	11.40	11.43	13.87	16.31	15.53	18.97
ship08l	14	14	12	12	11	11	11	11	10
ship08l	19.24	23.76	25.23	29.63	31.76	36.29	39.47	46.25	45.69
ship08s	14	13	12	11	10	11	10	10	10
ship08s	10.10	11.26	13.63	14.91	14.81	19.45	19.46	22.24	25.28
ship12l	18	15	13	13	13	13	15	12	13
ship12l	31.85	34.21	36.93	43.35	50.07	57.63	73.91	68.81	82.38
ship12s	16	14	14	13	13	12	12	12	12
ship12s	12.99	15.68	18.77	19.89	24.01	25.83	27.80	31.71	36.44
sierra	20	18	16	16	14	14	14	13	13
sierra	54.36	58.94	65.50	62.23	61.26	74.31	79.56	90.81	96.69
stair	16	13	12	13	13	13	12	11	11
stair	16.01	14.92	12.90	17.12	16.29	17.53	17.73	17.57	18.85
standata	13	11	13	13	11	11	10	11	11
standata	6.09	6.74	9.56	12.04	11.43	13.27	14.62	17.05	19.09
standmps	21	17	18	17	16	17	16	14	14
standmps	11.75	12.31	15.54	17.33	18.94	23.11	24.78	24.15	27.11
stocfor1	16	15	15	13	12	12	13	11	11
stocfor1	1.20	1.34	1.70	1.74	1.94	2.17	2.68	2.55	2.86
stocfor2	24	22	19	18	17	18	19	18	18
stocfor2	45.65	50.77	52.37	57.85	58.65	69.68	85.50	87.14	96.58
tuff	19	19	16	16	15	15	15	15	16
tuff	13.83	15.62	15.18	16.54	17.34	18.87	20.61	22.67	25.67
vtp.base	18	18	19	18	18	14	14	18	14
vtp.base	1.24	1.81	2.39	2.64	3.15	2.89	3.32	4.86	4.31
wood1p	28	27	29	21	23	18	27	22	22
wood1p	130.58	125.36	143.82	115.81	132.20	112.80	183.00	154.80	161.54
woodw	29	26	24	24	22	22	23	22	21
woodw	149.17	143.36	141.38	166.39	169.37	186.52	212.86	216.50	225.05
Total	1788	1668	1519	1478	1392	1357	1369	1339	1317
cpu(sec)	7071	7246	7221	7593	8024	8065	8977	9214	9835

Table 3.3: Performance of Algorithms II (Scr-Z).

References

[1] I. Adler, N. Karmarkar, M.G.C Resende, and G. Veiga (1989). "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming,* 44, 297-336.

[2] I. Adler, N. Karmarkar, M.G.C. Resende and G. Veiga (1989). "Data structures and programming techniques for the implementation of Karmarkar's algorithm." *ORSA Journal on Computing* 1(2), 84-106.

[3] R. Bixby (1990). "Implementing the Simplex Method: The Initial Basis," Technical Report, TR90-32, Rice University, Huston, Texas.

[4] 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 Technical Journal*, 68(3), 7-19.

[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. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD 20899.

[6] R. Fourer and S. Mehrotra (1991). "Performance of an Augmented System Approach for Solving Least-Squares Problems in an Interior-Point Method for Linear Programming," Technical Report (in preparation), Department of Industrial Engineering and Management Sciences, Northwestern University.

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

[8] N. Karmarkar, J.C. Lagarias, L. Slutsman and P. Wang (1989). "Power Series Variants of Karmarkar-type algorithms," AT&T Technical Journal, 68(3), 20–36.

[9] M. Kojima, S. Mizuno and A. Yoshise (1989). "A primal-dual interior point algorithm for linear programming," *Progress in Mathematical Programming, Interior Point and Related Method* (Springer-Verlag, New York) 29-47.

[10] M. Kojima, S. Mizuno and A. Yoshise (1988). "An $O(\sqrt{n}L) - iteration$ potential reduction algorithm for linear complementarity problems," Research reports on information sciences B-217, Dept. of information sciences, Tokyo institute of technology, Meguro-ku, Tokyo, Japan. (To appear in *Mathematical Programming*)

[11] 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, Gerogia Institute of Technology, Atlanta, GA 30332.

[12] Lustig, I., R. Marsten and D.F. Shanno (1990). "On Implementing Mehrotra's Predictor-Corrector Interior Point Method for Linear Programming," Technical Report SOR 90-03, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ 08544.

[13] S. Mehrotra (1990) "On the implementation of a primal-dual interior point method." Technical Report 90-03R, Department of Industrial Engineering and Management Sciences, Northwestern University (1990).

[14] S. Mehrotra (1991). "Handling free variables in interior methods." Technical Report 91-06, Department of Industrial Engineering and Management Sciences, Northwestern University.

[15] S. Mehrotra and Y. Ye (1991). "On Finding the Optimal Facet of Linear Programs." Technical Report 91-10, Department of Industrial Engineering and Management Sciences, Northwestern University.

[16] N. Megiddo (1986). "Pathways to the Optimal Set in Linear Programming," In *Progress in Mathematical Programming* Springer-Verlag, New York, 131-158.

[17] R.C. Monteiro and I. Adler (1989). "An $O(n^3L)$ primal-dual interior point algorithm for linear programming," *Mathematical Programming,* 44, 43–66, 1989.

[18] 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," *Mathematics of Operations Research,* 15(2), 191–214.