Efficient Implementation and Benchmark of Interior Point Methods for the Polynomial L_1 Fitting Problem

Aurelio R. L. Oliveira[1]

School of Electrical Engineering and Computer Science (FEEC), State University of Campinas (UNICAMP), R. Albert Einstein s/n, 13083-970 Campinas, SP, Brazil; aurelio@densis.fee.unicamp.br

Mário A. Nascimento[2]

Department of Computing Science, University of Alberta, Edmonton, AB, T6G 2H1, Canada; nascimento@computer.org

Christiano Lyra

School of Electrical Engineering and Computer Science (FEEC), State University of Campinas (UNICAMP), R. Albert Einstein s/n, 13083-970 Campinas, SP, Brazil; chrlyra@densis.fee.unicamp.br

Abstract

Interior point methods specialized to the L_1 fitting problem are surveyed and the affine scaling primal method is presented. Their main features are highlighted and improvements are proposed for polynomial fitting problems. For such problems, a careful handling of data avoids storing of matrices for the interior point approaches. Moreover, the computational complexity of iterations is reduced. An inexpensive way to compute a basic solution, using interpolation, is also provided. Extensive numerical experiments are carried out, including comparisons with a specialized simplex method. In general, the interior point methods performed better than the simplex approach. Among the interior point methods investigated, the dual affine scaling version was the most efficient.

Key words: L_1 curve fitting, L_1 regression problems, L_1 based statistical analysis, interior point methods, linear programming, data mining.

1 Introduction

L_1-norm based curve fitting is a recognized robust method for data analysis in statistics. Its main strength is the ability to diminish effect of outliers upon estimates; other relevant aspects are appraised by Dodge (1987b). Recently, statistics techniques have found their way into areas previously considered unrelated, for example, the one of databases and data mining in particular (Glymour, Madigan, Pregibon and Smyth, 1996; Elder and Pregibon, 1995). In data mining one may be interested in obtaining interesting patterns from a large dataset (e.g., purchase transactions in a department store), as well as in predicting behavior (e.g., variables) (Berson and Smith, 1997). Therefore, efficient solutions for curve fitting problems in general, and the L_1 curve fitting problem in particular, are of current interest.

Linear programming provides a convenient representation for the L_1 fitting problem. This representation, known since late fifties, brought along a practical way to solve this class of problems — namely, the simplex algorithm; it played a major role in the wide use of L_1-norm based statistical data analysis.

The first specialized linear programming methods for solving L_1 fitting problems to appear are variants of the simplex algorithm (Wagner, 1959; Barrodale and Roberts, 1973; Armstrong, Frome and Kung, 1979; Abdelmalek, 1980). Since the advent of the interior point approach for linear programming in mid eighties (Karmarkar, 1984), a new family of specially designed interior point methods has been developed (Meketon, 1987; Sherali, Skarpness and Kim, 1988; Ruzinsky and Olsen, 1989; Zhang, 1993). Another technique closely related to the interior point family was proposed by Coleman and Li (1992b).

This work revisits several interior point methods for the L_1 fitting problem proposed in the past ten years. The polynomial L_1 fitting problem is discussed in more detail; by exploiting its special structure we propose improvements that enhance all interior point approaches to the problem. These improvements lead to reduction of the computational complexity and memory requirements of the interior point methods. We also show how a basic solution can be easily obtained for the polynomial fitting problem once the interior point method terminates.

Computational performance of interior point and Coleman and Li's (1992b) methods for L_1 fitting problems are benchmarked. They are all compared against one of the best known implementation of the specialized simplex ap-

[1] Supported by the Research Supporting Foundation of the State of São Paulo (FAPESP), Brazil.
[2] Research performed while at the Brazilian Agency for Agricultural Research (EMBRAPA), Brazil.

proach for this problem (Armstrong et al., 1979).

This paper is organized as follows. Next section discusses the main interior point methods for the L_1 fitting problem. Section 3 brings improvements to the polynomial fitting problem. Extensive numerical results are presented in Section 4. Conclusions follow.

Notation. We use the following notation throughout this work. Lower case Greek letters denote scalars, lower case Latin letters denote vectors and upper case Latin letters denote matrices. Components of vectors are represented by the corresponding letter with subscripts. The symbol 0 will denote the scalar zero, the zero column vector and the zero matrix; its dimension will be clear from context. The identity matrix will be denoted by I. The Euclidean norm is represented by $\|\cdot\|$. Other p-norms are represented by $\|\cdot\|_p$. Diagonal matrices whose diagonal entries are the components of a vector represented by the same Latin letter (in lower case), $X = diag(x)$ for instance, are used throughout the paper.

2 Interior Point Methods for the L_1 Fitting Problem

Consider the L_1 fitting problem,

$$\begin{aligned} &\text{minimize} \quad \|r\|_1 \\ &\text{subject to} \quad Ax + r = b \end{aligned} \tag{1}$$

where A is a full column rank $n \times m$ matrix and r, b and x are column vectors of appropriate dimension. It is well known that this problem can be transformed into a linear programming problem by writing r as the difference between two nonnegative variables $r = u - v$

$$\begin{aligned} &\text{minimize} \quad e^t(u+v) \\ &\text{subject to} \quad Ax + u - v = b,\ (u,v) \geq 0, \end{aligned} \tag{2}$$

where e is the n-vector of all ones.

Associated with problem (2) is the dual linear programming problem

$$\begin{aligned} &\text{maximize} \quad b^t y \\ &\text{subject to} \quad A^t y = 0,\ -e \leq y \leq e, \end{aligned} \tag{3}$$

where y is a n-vector. The optimality conditions for (2) and (3) is given by

$$\begin{pmatrix} Ax + u - v - b \\ A^t y \\ p + y - e \\ q - y - e \\ UPe \\ VQe \end{pmatrix} = 0, \quad (u, v, p, q) \geq 0, \tag{4}$$

where p and q are vectors of slack variables (for y). Recall that U,V,P and Q are diagonal matrices whose diagonal entries are, respectively, the components of u, v, p, and q.

The dual affine-scaling method[3] (Meketon, 1987) was the first specialization of interior point methods for solving (2) to appear. The same method with a different choice of the scaling matrix is presented in (Ruzinsky and Olsen, 1989). This work adopts the second choice since it uses the nowadays standard way for dealing with bounded variables in interior point methods. A primal-dual method was developed by Zhang (1993) together with a predictor-corrector variant. The primal-dual and the predictor-corrector are compared against a specialized simplex implementation in (Duarte and Vanderbei, 1994). In order to complete this family of methods, we present the primal affine-scaling method.

The main ideas of the the method described by Coleman and Li (1992b) are pretty much in the flavor of interior point techniques. In particular, its iterations are similar to the iterations of interior point approaches; the most noticeable difference being the line search procedure. The quadratic convergence result shown for this method leaded us to study it.

Figures 1, 2, 3 and 4 summarize the interior point approaches specialized for L_1 fitting problems (2). Figure 5 summarizes the method proposed by Coleman and Li (1992b). In the description of the methods we sometimes use the notation $z = (x, u, v)$ and the gap $\gamma = u^t p + v^t q$. The parameters τ and ρ used in some of the methods are discussed in the Numerical Experiments section. The convergence criteria and starting points are also discussed later.

It is convenient to define $2p = e - y$ and $2q = e + y$ for the implementation of the dual method. However, these definitions were not used in Figure 2 to achieve

[3] By dual method we mean a method that solves the related dual problem in order to obtain the solution of the primal problem.

Given x^0 and $(u^0, v^0) > 0$ such that $Ax^0 + u^0 - v^0 = b$.
For $k = 0, 1, 2, \ldots$, do

$$\begin{aligned}
D^k &= ((U^k)^2 + (V^k)^2)^{-1} && (1)\\
w^k &= D^k((U^k)^2 - (V^k)^2)e && (2)\\
\Delta x^k &= (A^t D^k A)^{-1} A^t w^k && (3)\\
y^k &= w^k - D^k A \Delta x^k && (4)\\
\Delta u^k &= -(U^k)^2 p^k && (5)\\
\Delta v^k &= -(V^k)^2 q^k && (6)\\
\alpha^k &= \tau \min_i \left\{ \min \left\{ \frac{-u_i^k}{\Delta u_i^k} \middle| \Delta u_i^k < 0 \,;\, \frac{-v_i^k}{\Delta v_i^k} \middle| \Delta v_i^k < 0 \right\} \right\} && (7)\\
z^{k+1} &= z^k + \alpha^k \Delta z^k && (8)
\end{aligned}$$

Until Convergence.

Fig. 1. Primal Affine-Scaling Method (L1P)

Given $-e < y^0 < e$ such that $A^t y^0 = 0$.
For $k = 0, 1, 2, \ldots$, do

$$\begin{aligned}
D^k &= \frac{(P^k Q^k)^2}{4}((P^k)^2 + (Q^k)^2)^{-1} && (1)\\
x^k &= (A^t D^k A)^{-1} A^t D^k b && (2)\\
r^k &= b - Ax^k && (3)\\
\Delta q^k &= D^k r^k && (4)\\
\Delta p^k &= -\Delta q^k && (5)\\
\alpha^k &= \tau \min_i \left\{ \min \left\{ \frac{-p_i^k}{\Delta p_i^k} \middle| \Delta p_i^k < 0 \,;\, \frac{-q_i^k}{\Delta q_i^k} \middle| \Delta q_i^k < 0 \right\} \right\} && (6)\\
p^{k+1} &= p^k + \alpha^k \Delta p^k && (7)\\
q^{k+1} &= q^k + \alpha^k \Delta q^k && (8)\\
y^{k+1} &= \frac{q^k - p^k}{2} && (9)
\end{aligned}$$

Until Convergence.

Fig. 2. Dual Affine-Scaling Method (L1D)

a cleaner notation. Zhang (1993) works with a slight different formulation of the linear programming problem. Figures 3 and 4 present the version of his methods for problem (2). Because we noticed some convergence difficulties when $\mu^k = \left(\frac{\gamma^k}{2n}\right)^2$, as proposed by Zhang, this perturbation parameter was changed to $\mu^k = \left(\frac{\gamma^k}{(2n)^{\frac{3}{2}}}\right)$.

Given x^0 and $(u^0, v^0) > 0$ such that $Ax^0 + u^0 - v^0 = b$ and $-e < y^0 < e$ such that $A^t y^0 = 0$.
For $k = 0, 1, 2, \ldots$, do

$$\mu^k = \frac{\gamma^k}{(2n)^{\frac{3}{2}}} \tag{1}$$

$$D^k = (U^k(P^k)^{-1} + V^k(Q^k)^{-1})^{-1} \tag{2}$$

$$w^k = D^k(\mu^k((Q^k)^{-1} - (P^k)^{-1})e + u^k - v^k) \tag{3}$$

$$\Delta x^k = (A^t D^k A)^{-1} A^t w^k \tag{4}$$

$$\Delta y^k = w^k - D^k A \Delta x^k \tag{5}$$

$$\Delta u^k = \mu^k (P^k)^{-1} e - u^k + U^k (P^k)^{-1} \Delta y^k \tag{6}$$

$$\Delta v^k = \mu^k (Q^k)^{-1} e - v^k - V^k (Q^k)^{-1} \Delta y^k \tag{7}$$

$$\hat{\alpha}_p^k = \min_i \left\{ \min \left\{ \frac{-u_i^k}{\Delta u_i^k} \middle| \Delta u_i^k < 0\,;\, \frac{-v_i^k}{\Delta v_i^k} \middle| \Delta v_i^k < 0 \right\} \right\} \tag{8}$$

$$\hat{\alpha}_d^k = \min_i \left\{ \min \left\{ \frac{-q_i^k}{\Delta y_i^k} \middle| \Delta y_i^k < 0\,;\, \frac{p_i^k}{\Delta y_i^k} \middle| \Delta y_i^k > 0 \right\} \right\} \tag{9}$$

$$(\alpha_p^k, \alpha_d^k) = \begin{cases} (\hat{\alpha}_p^k, \hat{\alpha}_p^k), & \text{if } \hat{\alpha}_d^k > \hat{\alpha}_p^k \text{ and } u_k^t \Delta p^k + v_k^t \Delta q^k > 0, \\ (\hat{\alpha}_d^k, \hat{\alpha}_d^k), & \text{if } \hat{\alpha}_p^k > \hat{\alpha}_d^k \text{ and } p_k^t \Delta u^k + q_k^t \Delta v^k > 0, \\ (\hat{\alpha}_p^k, \hat{\alpha}_d^k), & \text{otherwise.} \end{cases} \tag{10}$$

$$z^{k+1} = z^k + \tau^k \alpha_p^k \Delta z^k \tag{11}$$

$$y^{k+1} = y^k + \tau^k \alpha_d^k \Delta y^k \tag{12}$$

Until Convergence.

Fig. 3. Primal-Dual Method (L1PD)

The line search procedure (that is, the computation of the step length) in Step (8) of the method, shown in Figure 5, is computed by looking at all $\alpha_i^k = -r_i^k / \Delta r_i^k > 0$. Observe that this procedure is $O(n^2)$ in the worst case. In Coleman and Li (1992b) the step length is given by

$$\alpha^k = \alpha_\#^k + \nu(\alpha_*^k - \alpha_\#^k),$$

where $\alpha_\#^k = \max\{0; \max_i \alpha_i^k | 0 < \alpha_i^k < \alpha_*^k\}$ instead of $\tau\alpha_*^k$. However, we adopted the step length shown in Steps (9) and (10) of Figure 5, since we experienced numerical problems in some test problems when α_*^k and $\alpha_\#^k$ are too close together.

Notice that there is a related method, for each approach, to the L_∞ fitting problem. The dual affine-scaling version was presented by Ruzinsky and Olsen (1989). A primal-dual and a predictor-corrector variant were presented by Zhang (1993). Coleman and Li (1992a) also presented a related variant for

Given x^0 and $(u^0, v^0) > 0$ such that $Ax^0 + u^0 - v^0 = b$ and $-e < y^0 < e$ such that $A^t y^0 = 0$.
For $k = 0, 1, 2, \ldots$, do

$$\mu^k = \frac{\gamma^k}{(2n)^{\frac{3}{2}}} \tag{1}$$

$$D^k = (U^k (P^k)^{-1} + V^k (Q^k)^{-1})^{-1} \tag{2}$$

$$\Delta\tilde{x}^k = (A^t D^k A)^{-1} A^t D^k (u^k - v^k) \tag{3}$$

$$\Delta\tilde{y}^k = D^k (u^k - v^k - A\Delta\tilde{x}^k) \tag{4}$$

$$\Delta\tilde{u}^k = -u^k + U^k (P^k)^{-1} \Delta\tilde{y}^k \tag{5}$$

$$\Delta\tilde{v}^k = -v^k - V^k (Q^k)^{-1} \Delta\tilde{y}^k \tag{6}$$

$$\begin{aligned} w^k = {} & D^k (u^k - v^k + (\mu^k I - \Delta\tilde{Y}^k \Delta\tilde{V}^k)(Q^k)^{-1} e - \\ & (\mu^k I + \Delta\tilde{Y}^k \Delta\tilde{U}^k)(P^k)^{-1} e) \end{aligned} \tag{7}$$

$$\Delta x^k = (A^t D^k A)^{-1} A^t w^k \tag{8}$$

$$\Delta y^k = w^k - D^k A \Delta x^k \tag{9}$$

$$\Delta u^k = (\mu^k I + \Delta\tilde{Y}^k \Delta\tilde{U}^k)(P^k)^{-1} e - u^k + U^k (P^k)^{-1} \Delta y^k \tag{10}$$

$$\Delta v^k = (\mu^k I - \Delta\tilde{Y}^k \Delta\tilde{V}^k)(Q^k)^{-1} e - v^k - V^k (Q^k)^{-1} \Delta y^k \tag{11}$$

$$\hat{\alpha}_p^k = \min_i \left\{ \min \left\{ \frac{-u_i^k}{\Delta u_i^k} \middle| \Delta u_i^k < 0 \, ; \frac{-v_i^k}{\Delta v_i^k} \middle| \Delta v_i^k < 0 \right\} \right\} \tag{12}$$

$$\hat{\alpha}_d^k = \min_i \left\{ \min \left\{ \frac{-q_i^k}{\Delta y_i^k} \middle| \Delta y_i^k < 0 \, ; \frac{p_i^k}{\Delta y_i^k} \middle| \Delta y_i^k > 0 \right\} \right\} \tag{13}$$

$$(\alpha_p^k, \alpha_d^k) = \begin{cases} (\hat{\alpha}_p^k, \hat{\alpha}_p^k), & \text{if } \hat{\alpha}_d^k > \hat{\alpha}_p^k \text{ and } u_k^t \Delta p^k + v_k^t \Delta q^k > 0, \\ (\hat{\alpha}_d^k, \hat{\alpha}_d^k), & \text{if } \hat{\alpha}_p^k > \hat{\alpha}_d^k \text{ and } p_k^t \Delta u^k + q_k^t \Delta v^k > 0, \\ (\hat{\alpha}_p^k, \hat{\alpha}_d^k), & \text{otherwise.} \end{cases} \tag{14}$$

$$z^{k+1} = z^k + \tau^k \alpha_p^k \Delta z^k \tag{15}$$

$$y^{k+1} = y^k + \tau^k \alpha_d^k \Delta y^k \tag{16}$$

Until Convergence.

Fig. 4. Predictor-Corrector Method (L1PC)

their method. We are not aware of any work proposing the primal affine-scaling version for this problem; nevertheless, it is not difficult to develop it from the general primal affine-scaling method (Vanderbei, 1989). The L_∞ problem will no longer be discussed here. However, it is worthy to remark that most of the ideas in this paper also apply to the specialized L_∞ interior point methods. They are subject of a forthcoming paper.

```
Given $x^0$, $r^0 = b - Ax^0$ and $-e < y^0 < e$ such that $A^t y^0 = 0$.
For $k = 0, 1, 2, \ldots$, do
    $w^k = sign(r^k)$    (1)
    $\beta^k = \max_i \left\{ \max \left\{ r_i^k \frac{(w_i^k - y_i^k)}{\|r^0\|_1}; |y_i^k| - 1 \right\} \right\}$    (2)
    $\theta^k = \frac{\beta^k}{(\rho + \beta^k)}$    (3)
    $D^k = |(R^k)^{-1}(W^k + (1 - \theta^k)Y^k)|$    (4)
    $\Delta x^k = (A^t D^k A)^{-1} A^t w^k$    (5)
    $\Delta r^k = -A \Delta x^k$    (6)
    $y^{k+1} = w^k + D^k \Delta r^k$    (7)
    $\|r^k + \alpha_*^k \Delta r^k\|_1 = \min_{\alpha_i^k > 0} \|r^k + \alpha_i^k \Delta r^k\|_1$    (8)
    $x^{k+1} = x^k + \tau \alpha_*^k \Delta x^k$    (9)
    $r^{k+1} = r^k + \tau \alpha_*^k \Delta r^k$    (10)
Until Convergence.
```

Fig. 5. Coleman-Li Method (L1CL)

2.1 Considerations about Computational Requirements

Since the constraint matrix of this application is usually full, we do not have any concern about sparse structure in the following discussion. The most expensive step for all methods is the solution of a linear system with a matrix of type $A^t D^* A$, where D^* is a diagonal matrix whose diagonal entries depend upon the method. The most widely used way for solving the linear system is to compute the matrix and then its Cholesky factorization (Golub and Van Loan, 1996), since it is symmetric and positive definite. Computing the matrix involves $\frac{3nm(m+1)}{2}$ *flops*[4] because it is symmetric and D^* is already at hand. The Cholesky factorization costs about $\frac{m^3}{3}$ flops. The solution of the triangular systems requires $2m^2 - m$ flops. The number of multiplications for computing the matrix can be halved by storing the entries of the dot products among each pair of columns of A. However, this approach needs $\frac{nm(m+1)}{2}$ memory positions of storage. All of the methods also require at least one matrix-vector product involving both A and A^t. Each matrix-vector product costs $2mn$ flops.

In resume, the computational complexity of one iteration is $O(nm^2 + n^2)$ for the L1CL method and $O(nm^2)$ for the other methods (recall that $m \leq n$).

[4] Flop is a float point operation.

Notice that the predictor-corrector variant requires the solution of two linear systems per iteration using the same matrix. Therefore, given the Cholesky factorization from the first system, solving the second linear system costs only $2m^2 - m$ extra flops.

3 Specialization to Polynomial L_1 Fitting Problems

Let us consider the special case were the data is approximated by a polynomial of degree $d = m - 1$. In this situation, the constraint matrix has a very structured form,

$$V^t = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_1^{m-1} & \alpha_2^{m-1} & \cdots & \alpha_n^{m-1} \end{pmatrix}$$

which is called a rectangular *Vandermonde matrix.*

Moreover, the product $H = VD^*V^t$ has the special structure of a *Hankel matrix*; its entries are constant along its antidiagonals. That is,

$$H = \begin{pmatrix} \eta_1 & \eta_2 & \cdots & \eta_m \\ \eta_2 & \eta_3 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \eta_{2m-2} \\ \eta_m & \cdots & \eta_{2m-2} & \eta_{2m-1} \end{pmatrix}.$$

With a careful use of this structure, the computational complexity of an iteration can be reduced. The reason is that there are $O(m^2)$ algorithms for solving Hankel systems (e.g., Trench, 1965) and H can be obtained in $O(nm)$ flops since only $2m-1$ entries are computed. Thus, the worst case complexity of one iteration is reduced to $O(n^2)$ for the L1CL method and $O(nm)$ for the other methods. Observe that these changes do not affect the line search procedure of L1CL (Step (8) of Figure 5). Therefore, as n becomes much larger than m the performance of L1CL method may deteriorate faster than the other methods.

The so called *fast algorithms* (e.g., Heinig and Jankowski, 1990) have even better computational complexity for solving Hankel systems. However, these

```
$\omega_i \leftarrow x_m$
$\omega_i \leftarrow x_j + \alpha_i \omega_i$ (for j=m-1,...,1)
```

Fig. 6. Procedure for Computing $w = Vx$

algorithms typically have a large overhead which are worth using only for larger problems than the ones we expect to solve.

Perhaps a more surprising result, there is no need to store any matrix in all the methods studied, due to the special structure of V. The matrix-vector products Vx or V^ty can be computed at the usual amount of flops without storing V. The same applies for computing H from V. Also, since H has only $2m-1$ different entry values, it can be stored as a vector. Furthermore, it is possible to compute its inverse in $O(m^2)$ flops (Trench, 1965) and simultaneously solve the linear system. Thus, no matrix is stored in any procedure altogether. Notice that Trench's method can be applied because H is positive definite — thus all leading principal submatrices are nonsingular. The only method where it may be convenient (but not necessary) to store a matrix is the predictor-corrector variant. The lower half of H^{-1} could be stored for solving the second linear system faster.

A procedure for computing the matrix-vector products $w = Vx$ with $2n(m-1)$ flops is given in Fig. 6 — notice that $\omega_i = \sum_{j=1}^{m} \alpha_i^j x_j$. A procedure to obtain $z = V^ty$ with $2nm - n$ flops is given in Fig. 7 — notice that z can written as $z = \sum_{i=1}^{n} y^t(1, \alpha_i, \alpha_i^2, \ldots, \alpha_i^{m-1})$.

Observe that a general matrix-vector product costs $2nm$ flops. This cost can be reduced to $2nm-n$ flops considering that all the component of the first column of V are one. In this context, the cost of computing the values $(\alpha_i)^j$ must be also considered. It amounts to $n(m-2)$ flops. However, this computation is done only once, before starting the iterations. Therefore, the procedures given before not only avoid storing matrix V but actually are slightly less expensive than the straightforward approach.

Figure 8 gives a similar procedure to compute matrix H, where δ_i^* is the *ith*-diagonal entry of D^*. The procedure costs $4nm - 3n$ flops.

There is yet another improvement for the polynomial fitting problem. It is very easy to find a basis without storing any matrix as before. In order to compute it, it is enough to select the m smallest residuals $|r_i|$ and interpolate the corresponding points. As the values of α_i are all distinct, any set of columns of V will lead to a nonsingular matrix for the interpolation. After the interpolation, a simplex based code can be used to find an optimal basis, if desirable.

In our implementation, the m smallest residuals are found using the heap

```
$z \leftarrow 0$
For $i = 1, \ldots, n$ do
    $\sigma \leftarrow y_i$
    $z_1 \leftarrow z_1 + y_i$
        For $j = 2, \ldots, m$ do
            $\sigma \leftarrow \alpha_i \sigma$
            $z_j \leftarrow z_j + \sigma$
```

Fig. 7. Procedure for Computing $z = V^t y$

```
$H \leftarrow 0$
For $i = 1, \ldots, n$ do
    $\sigma \leftarrow \delta_i^*$
    $\eta_1 \leftarrow \eta_1 + \delta_i^*$
        For $j = 2, \ldots, 2m-1$ do
            $\sigma \leftarrow \alpha_i \sigma$
            $\eta_j \leftarrow \eta_j + \sigma$
```

Fig. 8. Procedure for Computing H

algorithm (Sedgewick, 1983). Since all we need is the m smallest entries, this algorithm is suitable for our application — because it does not need to sort the whole vector r. Its computational complexity is $O(nlog(n))$.

After finding the points, the interpolation is computed in $O(m^2)$ flops by divided differences (Golub and Van Loan, 1996). This cross-over procedure would be much more expensive in a general context, because the sorting step can lead to singular matrices. Thus, several matrix factorizations may be computed before getting a basis. Moreover, these factorizations cost about m^3 flops when no special structure is present.

4 Numerical Experiments

We have run a large number of experiments in order to compare the standard and specialized version of interior point methods (IPMs)[5] against the specialized revised simplex method proposed by Armstrong et al. (1979) — this

[5] As previously noted, Coleman and Li's (1992b) is not strictly an IPM.

method was chosen because previous experiences reported by Dodge (1987a) considered it to be the best one.

The simplex method, referred to as L1AFK, was implemented as it appears in the original paper (coded in FORTRAN) — only few changes were made in the dimension of arrays, in order to run larger problems. All experiments were run on a SUN SPARCstation-20 under SunOS 5.5., using IEEE standard double precision floating point arithmetic. The IPMs were implemented in C and compiled with GNU's GCC compiler. L1AFK was compiled with GNU's G77 compiler which is a front end for GCC with options to recognize FORTRAN 77 — that is, all the executable codes are generated by the same compiler.

The stopping criteria is based in the optimality conditions (4). It measures the relative dual infeasibility.

$$\begin{pmatrix} \frac{\|A^t y^k\|}{1+\|y^k\|+\|a\|} \\ \\ \frac{\|e-y^k-p^k\|}{1+\|y^k\|+\|a\|} \\ \\ \frac{\|e+y^k-q^k\|}{1+\|y^k\|+\|a\|} \end{pmatrix} \leq \epsilon$$

and the relative gap

$$\frac{\max_i\{\max(u_i^k p_i^k; v_i^k q_i^k)\}}{1+\gamma^k} \leq \epsilon.$$

Where $a = (\alpha_1, \ldots, \alpha_n)$. Thus, $\|a\|$ is an estimate of $\|A\|$. The stopping tolerance ϵ is the square root of machine precision (Dennis and Schnabel, 1996).

Starting points are computed based on an idea from Coleman and Li (1992b),

$$x^0 = (A^t A)^{-1} A^t b \tag{5}$$

$$r^0 = b - Ax^0 \tag{6}$$

$$u_i^0 = \begin{cases} \frac{\lambda+1}{2} r_i^0 & \text{if } r_i^0 > 0 \\ -\frac{\lambda-1}{2} r_i^0 & \text{Otherwise.} \end{cases} \tag{7}$$

$$v_i^0 = \begin{cases} \frac{\lambda-1}{2} r_i^0 & \text{if } r_i^0 > 0 \\ -\frac{\lambda+1}{2} r_i^0 & \text{Otherwise.} \end{cases} \tag{8}$$

$$y^0 = \frac{\kappa r^0}{\|r^0\|_\infty}. \tag{9}$$

Actually, in Coleman and Li (1992b) u and v are not defined. We are proposing this choice for them in order to satisfy the relation $u^0 + v^0 = |\lambda r^0|$. Thus, if λ is $O(1)$, both u^0 and v^0 are of the same order as r^0. Recall that, by definition, $u - v = r$. For L1P we use $\lambda = 2$. For L1PD and L1PC we choose $\lambda = n$, since these methods work better for starting points not too close to the boundaries. For all methods we set $\kappa = 0.975$.

Another option is to start with $y^0 = 0$, as in Meketon (1987). This starting point is attractive because y^0 is in the middle of the set of constraints $-e \leq y \leq e$. However, we adopted the choice of y^0 shown in (9), because it performed slightly better in our experiments.

The parameter τ was set to 0.95 for L1P and L1D, For L1PD and L1PC, $\tau^k = \max(0.95, 1 - \mu^k)$ and $\tau = 0.975$ in L1CL as adopted by Coleman and Li (1992b). The parameter ρ in Step (3) of Figure 5 was set to 0.99.

We are mainly interested in comparing the CPU time needed by each of the methods as a function of n (number of observations) and d (degree of the fitting polynomial) — recall that $d = m - 1$. Several data sets were randomly generated as follows: (*i*) n was fixed at 2500 and d varied ($d = 1, 2, 3$ and 5) — these data sets yielded the curves shown in Figure 9; (*ii*) d was set at 3 and n varied as 500, 1000, 2500 and 5000 — Figure 10 was obtained using results from such data set. The results shown are the average obtained upon several runs. Time to input and output data was not accounted.

Small values for d were adopted because we are not aware of real life applications which approximate polynomials of high degree. The values for the independent variable are n equally spaced points in the range [0,1]. The dependent variable was randomly generated using the uniform $U(0, 1)$, normal $N(0, 1)$ and exponential $Exp(1)$ distributions. Qualitatively, the results were not noticeably influenced by the statistical distribution of the dependent variable; hence, only results obtained with the $U(0, 1)$ distribution are reported.

From now on, we refer to the specialized (fast) IPMs as X-f, and the standard (slow) IPMs as X-s, where X is the acronym used for the methods (L1P, L1D, L1PD, L1PC, L1CL). We also included in the test a modification of L1CL which takes the full step size (α) along the direction; this approach reduces the computational complexity of the iteration making it the same as the interior point methods. We refer to this modification as L1CLF.

In order to ease visualization, all curves in Figures 9 and 10, other than L1CL, have the same range on the y-axis. Performance of the simplex method (L1AFK) was used as a yardstick.

Our main observation from Figures 9 and 10 can be summarized as follows:

- L1P-f was about 33% faster than L1P-s; in all cases except one, both lost to L1AFK. Their performances were highly non-uniform and L1P-f presented some problems of numerical instability as n grew.
- L1D-f (L1D-s) was nearly 43% (27%) faster than L1AFK. As either d or n grew L1D became more attractive.
- L1PD-f was up to 37% faster than L1PD-s. For very small values of n L1AFK performed better. However, as either d or n got larger (mainly n) L1PD-f became the clear winner.
- L1PC's behavior followed pretty much the same of L1PD, i.e., it became a better choice over L1AFK as d and n grew.
- L1CL was, by far, the slowest of all methods we tested. L1CL-f was between 2.6% and 11% faster than L1CL-s (being more advantageous as d grew). Increasing n did not seem to favor L1CL-f.
- L1CLF obtained much better results than L1CL. However, a drawback, not shown in the figures is the lost of robustness. It failed for 25% of the cases, mainly for problems of large degree.

Let us now briefly discuss the number of iterations required by each of the IPMs. The results reported on Table 1 are the average obtained upon all numerical experiments with randomly generated data. In most cases the standard implementation yielded the same number of iterations of the specialized one. In cases where this did not happen we used the largest value. Nonetheless, the difference was rarely larger than one. The only exception to this were L1P and L1CLF, where larger differences occurred, and where some of the cases run presented numerical problems. The number of iterations required by L1AFK is not shown because it is a method based on a completely different approach.

Table 1
Number of Iterations Required by the IPMs.

	Average	Minimum	Maximum
L1P	39.5	12	93
L1D	11.1	9	13
L1PD	10.7	9	15
L1PC	8.78	7	11
L1CL	8.5	6	14
L1CLF	16.8	6	30

Table 1 shows that L1D, L1PD, L1PC, and L1CL are efficient approaches with regard to the number of iterations. On the other hand, L1P not only requires more iterations to converge, but also shows a highly variable behavior (also reflected on Figures 9 and 10).

While it may appear at first glance from Table 1 that L1CL is a competitive

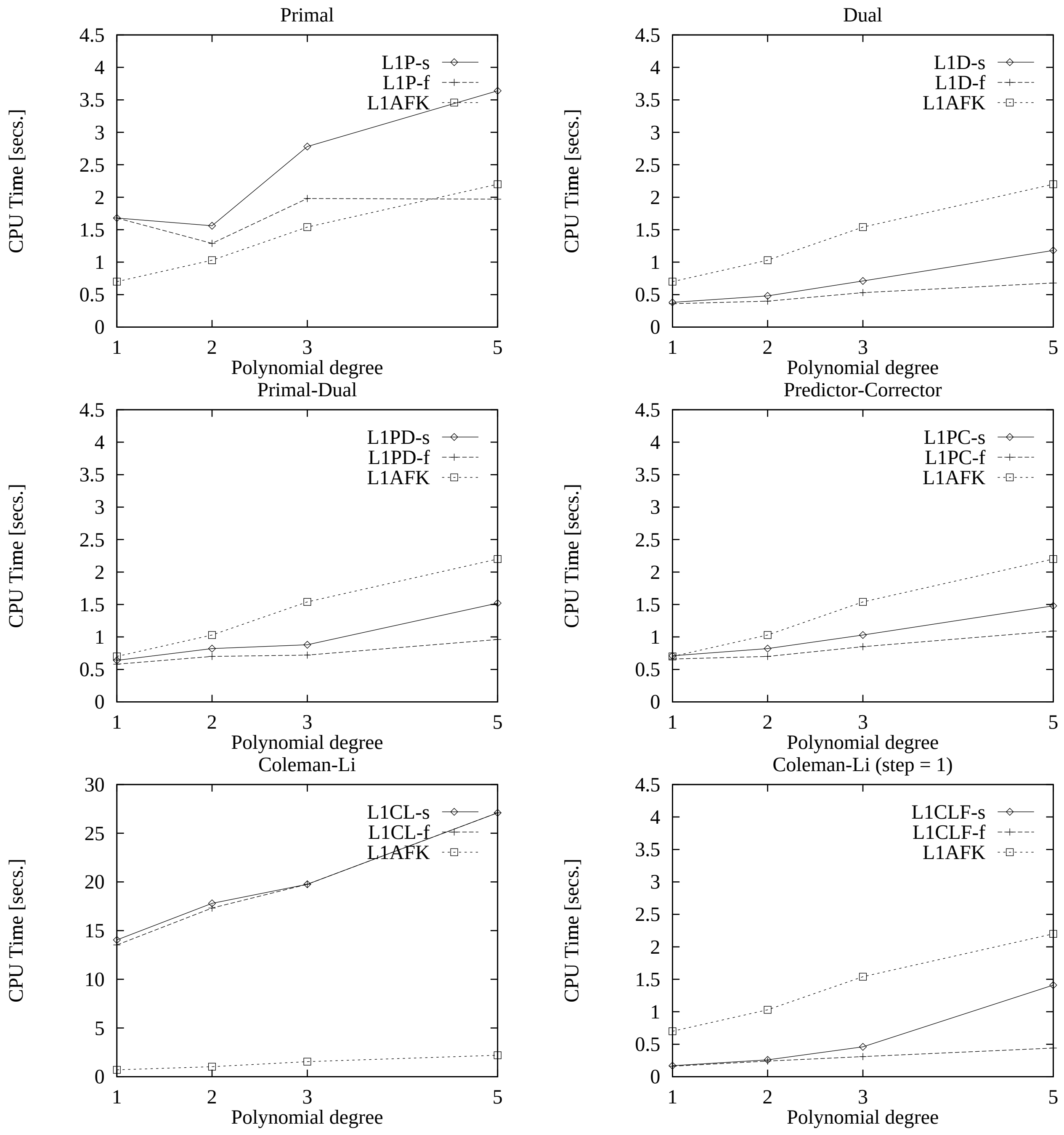

Fig. 9. CPU Time as a Function of d (degree of the fitting polynomial).

approach, one should keep in mind that it is computationally slow; it yields a low number of iterations, but each iteration is computationally expensive. This is due, as stressed earlier in this paper, to the fact that it employs an expensive line search ($O(n^2)$). One idea for future research is to explore other possibilities of line search procedures, such as interpolation, instead of the one proposed by Coleman and Li (1992b). We are aware that the theoretical results might be no longer valid. However, the savings on the computation of the step length may well compensate an eventual increment on the number of

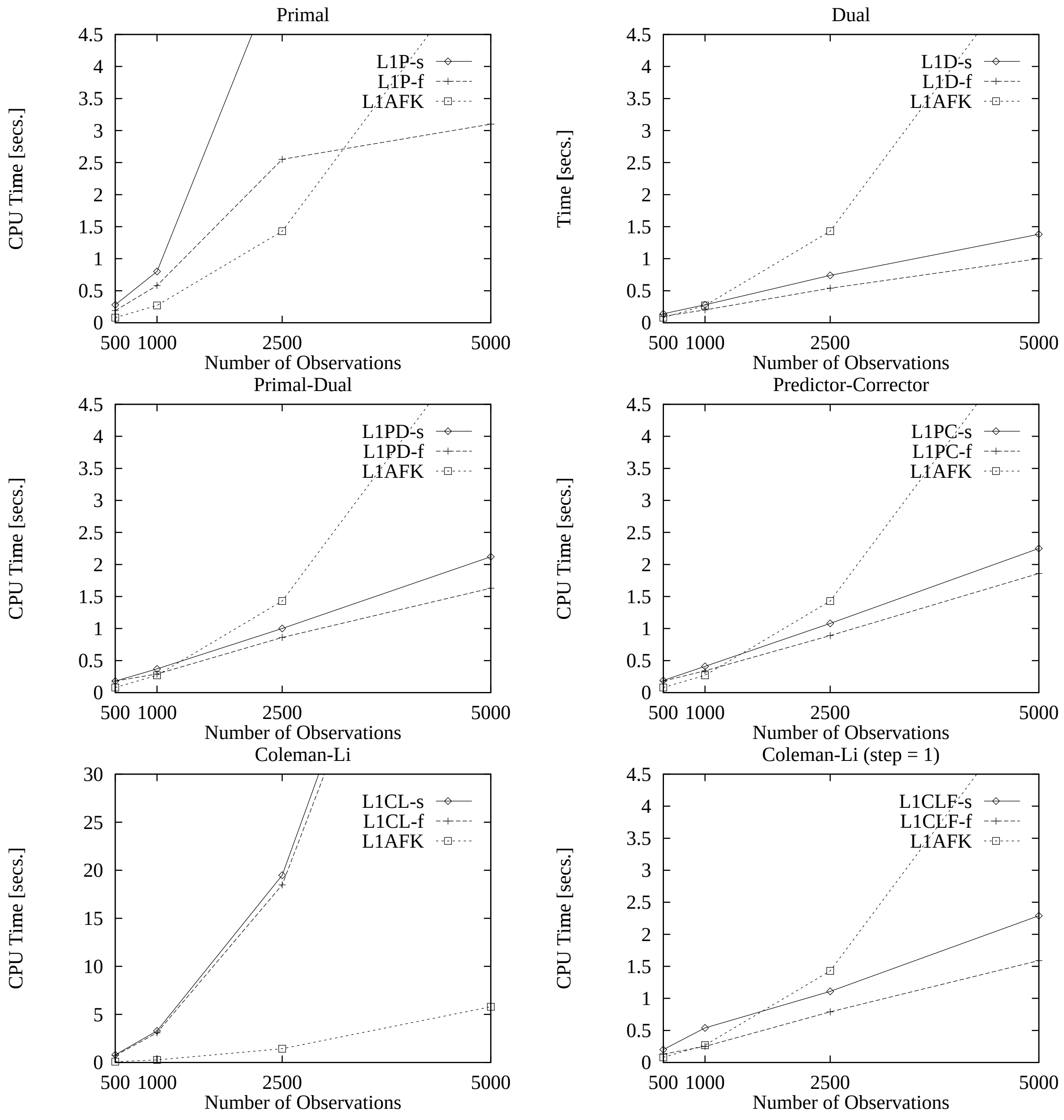

Fig. 10. CPU Time as a Function of n (number of observations).

iterations. The modification proposed here strengthen this assumption since when it works, it is very effective. Thus, another line search procedure should add robustness and yet be inexpensive.

Although L1D required a slight larger number of iterations compared to L1PD and L1PC, it obtained the smallest total running time — due to its less expensive iterations.

We have also run two experiments using (large) sets of real (financial) data[6]. The first data set contains the "prime rate"[7] (we refer to this set as PR). The second data set contains the "federal fund rate"[8] (we refer to it as FFR). Both data sets have daily (weekends and holidays not counted) values for over 40 years, totaling 10958 observed values. To avoid numerical problems we normalized the independent values (i.e., the observed dates) in the range [0,1]. The results obtained are shown in Tables 2(a) and (b). Results for L1CL and L1P are not presented, because the former took much more time to solve the problem (due to the large n) and the latter incurred in numerical instability. Finally, only the fast versions of the investigated IPMs were used.

Table 2
CPU Time [secs.] as a Function of the Degree d.

(a) Data set PR

	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
L1D	1.84	2.88	3.18	2.89	2.94
L1PD	10.88	15.79	14.94	12.29	7.90
L1PC	13.23	27.62	15.05	18.66	8.21
L1CLF	6.81	10.58	10.70	11.28	-
L1AFK	16.73	25.48	35.51	49.54	54.52

(b) Data set FFR

	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
L1D	1.45	2.64	3.83	3.19	3.35
L1PD	8.75	12.34	17.59	11.05	13.16
L1PC	7.12	17.76	18.68	8.39	11.46
L1CLF	7.42	7.39	9.33	9.89	12.40
L1AFK	14.21	20.30	29.82	37.51	42.79

From Tables 2(a) and (b) we can confirm two main conclusions: (*i*) L1D is indeed a very efficient approach; and (*ii*) L1AFK performance gets much worse than the IPMs as d increases. L1CLF did well, however it failed to converge for $d = 5$ in the PR data set.

With respect to the objective function's value, none of the approaches is con-

[6] Obtained at `http://www.bog.frb.fed.us/releases/h15/data.htm`, thanks to the indication of Lawrence A. Austen.
[7] The prime rate is a 7-day rate with weekends and holidays containing the prior business day's value.
[8] The federal funds rate is the cost of borrowing immediately available funds, primarily for one day.

sistently superior. For the same size of problems, the IPMs provided the best solution in some cases, whereas for others the L1AFK did so. This observation holds for both solutions computed by the IPMs, i.e., before the interpolation and after it. It should be mentioned that, for both implementations of L1D (slow and fast), the interpolation process provided a basis with a slightly worse value for the objective function than the one already available (non-basic). This did not occur in any other of the IPMs which obtain a slightly better value (an issue for future research). Despite that, we believe that L1D is, overall, the best option to solve the L_1 fitting problem.

Finally, we must remark that both L1PD and L1PC are very sensitive with respect to the starting point and the parameters. For instance, if we use $\lambda = 2$ in (7) as we do for L1P, the performance of these methods deteriorates significantly. On the other hand, a careful choice of the starting point and parameters could make one of them the method of choice for a given class of problems.

5 Conclusions

We reviewed in this paper the main interior point approaches to L_1 fitting problems: dual affine-scaling methods, primal-dual method, and its predictor-corrector variant. A new approach closely related to the interior point family, proposed by Coleman and Li (1992b), was also reviewed and a modification for computing the step length tested. Furthermore, the primal affine-scaling method was proposed.

Improvements were proposed for polynomial fitting problems, taking full advantage of its special structure. For such problems there is no need to store matrices and the computational complexity is reduced. Numerical experiments show that such improvements indeed lead to the expected reduction in computer requirements.

We believe that simplex approaches for the L_1 polynomial fitting problem do not permit such an elegant use of the problem structure as the one developed in this paper for interior point methods. This is indeed a surprising feature; usually simplex based methods lead to very good use of structures, while interior point methods enable only modest exploitations — network flow problems are a good example of the typical situation (Resende and Veiga, 1993).

The computation of H (Fig. 8) is the most expensive procedure of these methods. It is possible to implement this procedure in a parallel environment reducing the complexity of the iterations. In order to do that it would be necessary to store matrix the V. We were not tempted to follow this path due to the small computational times.

The improved dual affine-scaling approach (L1D-f) performed better than all other methods, for the extensive numerical experiments carried out. Some comments about the better performance of L1D-f are in order. The predictor-corrector variant is considered the best option for the general linear programming problem (e.g., Lustig, Marsten and Shanno, 1992). However, the problems solved in the general context are typically of larger dimension than the L_1 fitting problems. Moreover, the very special formulation of the dual problem (3) also deserves some comments.

(*i*) It allows the possibility to chose good starting points.
(*ii*) This formulation gives complete freedom with respect to the primal variables, since x and r are free in L1D (Figure 2) — what is perhaps a more important property.
(*iii*) Of all the methods studied, L1D-f is the one with smallest overhead on the iterations.

Comments (*i*) and (*ii*) explain the small number of iterations. Together with the small overhead, this help us to understand the somewhat surprising results obtained.

An important feature of all interior point methods for the L_1 fitting problem is that they are easy to implement. On the other hand, building specialized simplex methods requires much more work. L1D, in addition to being the best overall approach, is the easiest one to implement. Indeed, using programming tools such as *Matlab* and *Mathematica* its implementation can be a trivial task.

Acknowledgments

This research was partially sponsored by FAPESP (Research Supporting Foundation of the State of São Paulo, Brazil) and CNPq (Brazilian National Council for Scientific and Technological Development). Helpful comments from José Ruy Porto de Carvalho are also acknowledged.

References

Abdelmalek, N. N. (1980). l_1 solution of overdetermined systems of linear equations, *ACM Trans. on Mathematical Software* **6**(2): 220–227.

Armstrong, R. D., Frome, E. L. and Kung, D. S. (1979). A revised simplex algorithm for the absolute deviation curve fitting problem, *Commun. Statist.* **B8**: 175–190.

Barrodale, I. and Roberts, F. D. K. (1973). An improved algorithm for discrete l_1 linear approximation, *SIAM J. Numer. Anal.* **10**(5): 839–848.

Berson, A. and Smith, S. J. (1997). *Data Warehousing, Data Mining and OLAP*, McGraw Hill, Boston, MA.

Coleman, T. F. and Li, Y. (1992a). A global and quadratically convergent method for linear l_∞ problems, *SIAM J. Numer. Anal.* **29**(4): 1166–1186.

Coleman, T. F. and Li, Y. (1992b). A globally and quadratically convergent affine scaling method for linear l_1 problems, *Mathematical Programming* **56**: 189–222.

Dennis, J. E. and Schnabel, R. B. (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, PA.

Dodge, Y. (1987a). An introduction to l_1-norm based statistical data analysis, *Computational Statistics & Data Analysis* **5**(4): 239–253.

Dodge, Y. (ed.) (1987b). *Computational Statistics & Data Analysis*, Vol. 5(4), Special Issue on Statistical Data Analysis Based on the L_1 Norm and Related Methods, pp. 237–453.

Duarte, A. M. and Vanderbei, R. J. (1994). Primal-dual interior point algorithms for lsad and lmad estimation, *Technical report*, Princeton University.

Elder, J. F. and Pregibon, D. (1995). A statistical perspective on knowledge discovery in databases, *Proceedings of the 1st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 87–93.

Glymour, C., Madigan, D., Pregibon, D. and Smyth, P. (1996). Statistical inference and data mining, *Communications of the ACM* **30**(11): 35–41.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations Third Edition*, The Johns Hopkins University Press, Baltimore, Maryland.

Heinig, G. and Jankowski, P. (1990). Parallel and superfast algorithms for Hankel systems, *Numerische Mathematik* **58**: 109–127.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming, *Combinatorica* **4**(4): 373–395.

Lustig, I., Marsten, R. and Shanno, D. (1992). On implementing Mehrotra's predictor-corrector interior point method for linear programming, *SIAM J. Optimization* **2**: 435–449.

Meketon, M. S. (1987). Least absolute value regression, *Technical report*, AT&T Bell Laboratories.

Resende, M. G. C. and Veiga, G. (1993). An efficient impletation of a network interior point method, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **12**: 299–348.

Ruzinsky, S. A. and Olsen, E. T. (1989). l_1 and l_∞ minimization via a variant of Karmarkar's algorithm, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **37**(2): 245–253.

Sedgewick, R. (1983). *Algorithms*, Addison-Wesley, Reading, Massachusetts.

Sherali, H., Skarpness, B. and Kim, B. (1988). An assumption-free convergence analysis for a perturbation of the scaling algorithm for linear programs with application to the l_1 estimation problem, *Naval Res. Logist. Quart.* **35**: 473–492.

Trench, W. F. (1965). An algorithm for the inversion of finite Hankel matrices, *J. Soc. Indust. Appl. Math.* **13**(4): 1102–1107.

Vanderbei, R. J. (1989). Affine-scaling for linear programming with free variables, *Mathematical Programming* **43**: 31–44.

Wagner, W. H. (1959). Linear programming techniques for regression analysis, *J. Amer. Statist. Assoc.* **54**: 206–212.

Zhang, Y. (1993). A primal-dual interior point approach for computing l_1 and l_∞ solutions of overdetermined linear systems, *Journal of Optimization Theory and Applications* **77**(2): 323–341.