A Quadratically-Convergent Algorithm for the Linear Programming Problem with Lower and Upper Bounds*

Thomas F. Coleman†
Yuying Li‡

Abstract. We present a new algorithm to solve linear programming problems with finite lower and upper bounds. This algorithm generates an infinite sequence of points guaranteed to converge to the solution; the ultimate convergence rate is quadratic. The algorithm requires the solution of a linear least squares problem at each iteration - it is similar in this respect to recent interior point and "Karmarkar-like" methods. However, the algorithm does not require feasibility of the iterates; instead, monotonic decrease of an augmented linear l_1 function is maintained. A penalty parameter is not required. This method is particularily attractive for large-scale problems in that the number of iterations required to obtain high accuracy is relatively insensitive to problem size and is typically quite small. We provide results of numerical experiments.

1. Introduction. There is an intimate relationship between the linear l_1 problem and the linear programming problem . Indeed, linear l_1 problems can be formulated as linear programming problems; Bartels, Conn, and Sinclair [1] observe that provided a sufficiently large penalty parameter is introduced, a linear programming problem is a linear l_1 problem. Unfortunately, the threshold value of this parameter is not known a priori.

Here we show that a linear programming problem, with upper and lower bounds on all variables, is equivalent to an *augmented* linear l_1 problem - i.e., an l_1 function augmented with a linear term. This augmented form has no penalty parameter. The augmented linear l_1 function can be minimized by an algorithm similar to that proposed by Coleman and Li [3]. This method bears some similarity to interior point methods for linear programming in that a sequence of weighted linear least squares problems is solved and the number of iterations is relatively insensitive to the problem size; however, the algorithm differs in

* Presented at the Workshop on Large-Scale Numerical Optimization, Mathematical Sciences Institute, Cornell University. October 19, 1989.

† Computer Science Department and Center for Applied Mathematics, Cornell University, Ithaca, NY 14853. Research partially supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under grant DE-FG02-86ER25013.A000, by the U.S. Army Research Office through the Mathematical Sciences Institute, Cornell University, and by the Computational Mathematics Program of the National Science Foundation under grant DMS-8706133.

‡ Computer Science Department, Cornell University, Ithaca,NY 14853. Research partially supported by the U.S. Army Research Office through the Mathematical Sciences Institute, Cornell University and by the Computational Mathematics Program of the National Science Foundation under grant DMS-8706133.

two important ways. First, the sequence generated is ultimately quadratically convergent; second, the iterates are not feasible in general.

Consider the following linear programming problem:

$$\begin{aligned} & \min b^T\mu \\ \text{subject to} \quad & A\mu = c \\ & -e \le \mu \le e \end{aligned} \tag{1}$$

where e is a vector, of appropriate dimension, with each entry equal to unity, and A is an m-by-n matrix with $m < n$. Note: Any linear program with finite lower and upper bounds on all variables, $l \le \mu \le u$, can be expressed in this form after the appropriate scaling and translation. As we show below, (1) is equivalent to the minimization of an augmented l_1 function:

$$\min_{x \in \Re^m} \{-c^T x + \sum_{i=1}^{n} |a_i^T x - b_i|\}, \tag{2}$$

where a_i^T is row i of A^T. This equivalence can be seen by considering the dual of (1). Alternatively, consider the optimality conditions (e.g., [2]) for problem (2). Specifically, let $\mathcal{A}(x)$ denote the indices of zero residuals at any point x, i.e.,

$$\mathcal{A}(x) = \{\, i \mid a_i^T x - b_i = 0\} \tag{3}$$

and let $\mathcal{A}^c(x)$ denote the complementary set (we will suppress the argument when it is clear from context). Vector x is optimal to (2) if and only if there exists $\mu \in R^{|\mathcal{A}|}$ such that

$$-c + \sum_{i \in \mathcal{A}^c} \operatorname{sgn}(a_i^T x - b_i) a_i = -\sum_{i \in \mathcal{A}} \mu_i a_i \tag{4}$$

where

$$-1 \le \mu_i \le 1, \quad \forall i \in \mathcal{A}, \tag{5}$$

and sgn is defined as follows: if $w = sgn(v)$, where v is a vector, then

$$w_i = \begin{cases} 1 & \text{if } v_i \ge 0, \\ -1 & \text{otherwise.} \end{cases} \tag{6}$$

However, optimality conditions for (1) are well-known and can be expressed as follows: μ is optimal if and only if there exists $x \in R^m$ such that

$$-e \le \mu \le e \tag{7}$$

$$A\mu = c \tag{8}$$

$$b_i - a_i^T x \ne 0 \Rightarrow \mu_i = sgn(b_i - a_i^T x). \tag{9}$$

But (8) and (9) can be combined to yield a single condition:

$$c = \sum_{i \in \mathcal{A}^c} \operatorname{sgn}(b_i - a_i^T x) a_i + \sum_{i \in \mathcal{A}} \mu_i a_i, \tag{10}$$

where $\mu_i = sgn(b_i - a_i^T x)$ if $i \in A^c$. Conditions (7), (10) are clearly equivalent to (4), (5); therefore, the linear programming problem is equivalent to the augmented l_1 problem.

2. The Algorithm. In this section we briefly describe the augmented l_1 algorithm. It is similar in spirit to the algorithm derived in [3]; however, here we do not introduce an extra variable and the result is a simpler procedure. Moreover, if $c = 0$ then the proposed algorithm reduces to the l_1 algorithm proposed by Coleman and Li [4].

First we make a transformation of variables. Let Z denote a matrix whose rows form a basis for the null space of A. Assume $rank(A) = m$. Hence Z has dimensions $(n-m) \times n$, $rank(Z) = n - m$, and

$$AZ^T = 0.$$

Then, defining $r = b - A^T x$, problem (2) is equivalent to the following constrained l_1 problem with n variables r:

$$\min_{r \in \Re^n} \psi(r) \stackrel{def}{=} g_0^T r + \|r\|_1$$

$$\text{subject to} \quad Zr = Zb, \tag{11}$$

where $g_0 = A^T(AA^T)^{-1}c$. Optimality conditions for (11) can be expressed as: r is optimal if there exists $\lambda = Z^T w$, $w \in R^{n-m}$ such that

$$r_i(g_i - \lambda_i) = 0, \quad i = 1:n \tag{12}$$

$$Z(r - b) = 0 \tag{13}$$

$$-1 \leq \lambda_i - g_{0_i} \leq 1, \quad i \in \mathcal{A}, \tag{14}$$

where $g = g_0 + sgn(r)$. If (r, λ) satisfy (12)-(14) then it is easy to show that $x = (AA^T)^{-1}A(b - r)$ and $\mu = g_0 - \lambda$ satisfy (4),(5).

Define $\theta = \theta(r, \lambda)$ by [1]

$$\theta = \frac{\frac{\sqrt{\sum_{i=1}^n (r_i(g_i - \lambda_i))^2}}{\|r^0\|_\infty} + e^T \max\{|\lambda - g_0| - e, 0\}}{\rho + \frac{\sqrt{\sum_{i=1}^n (r_i(g_i - \lambda_i))^2}}{\|r^0\|_\infty} + e^T \max\{|\lambda - g_0| - e, 0\}}, \tag{15}$$

where $0 < \rho < 1$. Clearly $\theta^k = \theta(r^k, \lambda^k) \to 0$ if and only if $\{r^k, \lambda^k\} \to (r^*, \lambda^*)$ provided $Z(r^k - b) = 0$. Notice also that $\leq \theta < 1$. Of course there are many possible alternative definitions of θ which could satisfy the properties mentioned above; we have chosen the above definition because it is simple and appears to perform well in practise.

Define a diagonal scaling matrix as follows:

$$D^2 = |D_r D_\theta^{-1}| \tag{16}$$

where $D_r = \text{diag}(r)$, $D_\theta = \text{diag}(sgn(r) + (1 - \theta)(g_0 - \lambda))$, and $0 \leq \theta \leq 1$.

We can now state the algorithm. Let r^0 be an initial differentiable point satisfying $r_i^0 \neq 0$, $i = 1:m, Z(r^0 - b) = 0$; $k \leftarrow 0$; compute an initial point λ^0.

[1] Notation: If v is a vector and $y = max(v, 0)$ then y is a vector with components $y_i = v_i$ if $v_i > 0$; otherwise, $y_i = 0$. Similarily, if w is a vector and $z = |w|$ then z is a vector with components $z_i = |w_i|$.

Algorithm

Step 1 Compute θ^k from (15). Define D_k from (16) and $g^k \leftarrow \nabla\psi(r^k)$.

Step 2

$$\text{Solve } D_k^{-1}A^T d_x^k \overset{\text{l.s.}}{=} D_k g^k$$

$$d^k \leftarrow -A^T d_x^k, \quad \lambda^{k+1} \leftarrow g^k + D_k^{-2} d^k.$$

Step 3 Do a line search on the piecewise linear function $\psi(r^k + \alpha d^k)$ (as described below) to determine α^k,

$$r^{k+1} \leftarrow r^k + \alpha^k d^k, \quad k \leftarrow k+1.$$

Line Search: We first determine all nonnegative breakpoints. For each i with $d_i \neq 0$ let $\alpha_i = -\frac{r_i}{d_i}$ and let $J = \{i : \alpha_i > 0\}$. Then determine the minimizer in direction d:

$$\alpha_* = \arg\min_{\alpha>0} \psi(r+\alpha d) = \arg\min_{\alpha\in J} \psi(r+\alpha d).$$

This is done by considering each breakpoint in J in turn, adjusting the gradient to reflect a step just beyond the breakpoint and then determining if d continues to be a descent direction for ψ. For example, if α_j is the smallest positive breakpoint, then a step just beyond this point yields the following gradient:

$$g^+ = g - 2g_j e_j.$$

If $(g^+)^T d < 0$ the next breakpoint is considered, etc. Of course we cannot step all the way to the minimizer (since points of nondifferentiability must be avoided) and therefore we compute $\tilde{\alpha} = \max_{i\in J\cup\{0\}}\{\alpha_i : 0 \le \alpha_i < \alpha_*\}$, where $\alpha_0 = 0$; the steplength α is computed by

$$\alpha = \tilde{\alpha} + \max\{\tau, 1-\theta\}(\alpha_* - \tilde{\alpha}), \tag{17}$$

where $0 < \tau < 1$.

Remarks: 1. Based on Theorem 1 in [4], it is easy to see that if $\theta^k > 0$ then D_k is a positive diagonal matrix; therefore, d^k is a descent direction for ψ at r^k.
2. The computation of α ensures $\alpha^k \to 1$ as $k \to \infty$.
3. The dominant work is the least squares solve in Step 2.
4. Search direction d^k has the following property: as $\theta^k \to 0$, d^k approaches a Newton step based on the optimality conditions (12,13) (see [4] and [3] for more detail).

3. Convergence Properties. In [4], global and quadratic convergence properties are established for the linear l_1 algorithm. It is straightforward to modify each convergence result in [4] to the algorithm described here for the augmented l_1-problem. Indeed, replace each reference to $\|r\|_1$ (in [4]) with $\psi(r)$, as defined above, and keep in mind that $g^k = g_0 + sgn(r^k)$. There is no other change.

Before stating the result two definitions are needed.

DEFINITION 1. *We say an augmented l_1 problem is* **primal nondegenerate** *if and only if, at any point x the vectors a_i, $i \in \mathcal{A}(x)$, are linearly independent.*

DEFINITION 2. *We call* λ *a* **dual basic point** *if and only if* $A\lambda = c$ *and* $|\{\lambda_i : |\lambda_i| = 1\}| \geq n - m$. *We say an augmented* l_1 *problem is* **dual nondegenerate** *if and only if, at any dual basic point* λ, $|\{\lambda_i : |\lambda_i| = 1\}| = n - m$.

THEOREM 3.1. *Assuming both primal and dual nondegeneracy, the sequence* $\{(r^k, \lambda^k)\}$ *is convergent to the optimal pair* (r^*, λ^*). *Moreover, the ultimate rate of convergence is quadratic.*

4. **Numerical Results.** We have conducted some preliminary numerical experiments in order to obtain some indication as to whether there is practical potential in our method. We have generated (mostly) random, relatively small problems and compared our method to "standard" interior point approaches [5],[9].

The dependent variable θ^k is a measure of the distance from optimality: our stopping criterion is:

$$\theta^k < 10^{-13},$$

where machine precision on our system is approximately 10^{-16}.

The starting point (for New) is computed as follows: [2]

$$\begin{aligned} x^0 &\leftarrow \text{argmin } \{-\gamma c^T x + \|A^T x - b\|_2^2\} \\ r^0 &\leftarrow b - A^T x^0 \\ \lambda^0 &\leftarrow \frac{\tau}{\|r^0\|_\infty} r^0 \end{aligned}$$

The settings of the parameters are:

$$\tau \leftarrow .975, \quad \rho \leftarrow .99, \quad \gamma \leftarrow \sqrt{n}.$$

We have implemented the methods in PRO-MATLAB [6] using SUN 3/50 and 3/160 workstations. The linear least squares subproblems are solved with the orthogonal QR-factorization using row interchanges for greater stability [8]. No account was made of sparsity in our experiments.

If we set $c = 0$ then the algorithm reduces to the original linear l_1 procedure proposed in [4]. Numerical experiments are reported in [4]; here we give a small sample. In the following tables, "Dual" refers to the interior point algorithm proposed by Meketon [5] for l_1-problems. The origin is a natural starting point for this algorithm.

Function Approximation Problems

Determine $x = (\alpha_1, ..., \alpha_m)$ so that

$$\phi(z) = \sum_{j=1}^{m} \alpha_j \, z^{j-1},$$

is a best l_1 fit to $f(z)$ on the points $z = 0, \frac{1}{n}, ..., 1$.

[2] For simplicity we did not restrict λ^0 so that $\lambda^0 = Z^T w^0$ for some w^0. However, the computation of λ^i does enforce $\lambda^i = Z^T w^i$, $i = 1, 2, ...$

$m = 5, f(z) = \exp(z)$.

Number of Steps		
n	Dual	new
100	18	8
200	19	11
400	21	10
600	21	12

m = 6, f(z) = sin(z)

Number of Steps		
n	Dual	New
100	15	9
200	16	8
400	16	9
600	17	9

Next we consider randomly generated linear l_1 problems, i.e., $c = 0$.

n = 50

Number of Steps		
m	Dual	New
10	25	9
20	25	12
30	25	10
40	23	9

n = 100

Number of Steps		
m	Dual	New
10	25	10
20	26	13
40	26	13
50	26	11
70	25	11
90	23	10

On these problems the new algorithm outperforms the interior point method "Dual" by a factor of between two and three; both algorithms perform consistently and are fairly insensitive to problem size.

We have also generated more general linear programming problems, i.e., $c \neq 0$, subject to general upper and lower bounds: We used the MATLAB function *rand* to create matrix A and vectors b, c, ub. Then, $Ax = b$ was solved by computing the least-squares solution ("backslash"); columns of A and components of x were adjusted in sign to ensure $x > 0$. Finally, for each i,

$$ub_i \leftarrow \max\{2x_i, ub_i\}.$$

Our algorithm was applied after first transforming (translating and scaling) to achieve unit upper and lower bounds. We included the MINOS [7] package in our experiments - the number of iterations required by MINOS is also included in our table of results (we use the default parameter settings). Column "IP" refers to the number of iterations required by our MATLAB implementation of the interior point variant proposed in [9].

Random LP Problems:

n = 50

Number of Steps			
m	Minos	IP	New
10	58	32	11
20	85	27	11
30	109	29	11
40	99	26	9

n = 100

Number of Steps			
m	Minos	IP	New
10	56	38	11
30	168	31	18
40	228	30	15
60	272	29	12
70	223	29	16
90	194	31	13

n = 200

Number of Steps			
m	Minos	IP	New
10	194	63	19
20	249	35	17
30	325	32	20
50	490	30	15
70	568	31	18
100	651	30	15
140	600	30	14
160	714	32	14
190	421	27	14

Observations: Comparing "IP" to "New" we see that the new algorithm typically requires about half the number of iterations required by "IP". Since the dominant cost of each iteration is the same - equivalently structured least squares solutions - we conclude that the proposed algorithm is about twice as fast on this set of problems.

MINOS requires many more iterations than either of the other two methods; the number of iterations required by MINOS is more sensitive to the problem dimensions.

Note: Here we are not comparing the computational cost of MINOS to the other methods - we include the MINOS numbers to gain some indication of the dependancy of the simplex method on problem size. A meaningful comparison of computational costs would involve careful implementations and experiments on large-scale problems taking into account sparsity - we are not at that stage yet. However, the comparison of "IP"-iterations versus "New"-iterations does have some meaning since each iteration has roughly equivalent complexity.

5. Concluding Remarks. We have proposed a new algorithm for solving linear programming problems with lower and upper bounds on the variables. Theoretically, the algorithm possesses two attractive properties: global convergence and local quadratic convergence. Computationally, this method is similar to the interior point approach - a least squares problem with identical structure is solved at each iteration. However, our method does not require feasibility of the iterates. Our preliminary experiments indicate a significant improvement, in overall computational cost, compared to the interior point approach. Experiments on near-degenerate l_1 problems, reported in [4], indicate a degradation in performance on this class of problems. We expect similar behaviour here; this is a subject of continuing research.

A more convincing indicator of the practical potential of our proposed method will be provided be the outcome of numerical experiments with the standard collection of linear programming test problems. We have not done this yet for two reasons. First, a sparse implementation of our method is required: to date we have only a dense MATLAB implementation. Second, our proposed method requires explicit lower and upper bounds but almost all the standard test problems do not have this form. We are currently working on both these points and hope to report on our numerical experience soon.

Least-squares based methods, such as the proposed algorithm and interior point methods, have some attractive features for large-scale problems. In particular, they benefit from

any advance in the ability to solve least-squares problems. Since the (sparse) least-squares problem is a fundamental computation arising in many contexts, there is considerable on-going work on the development of efficient methods to perform least squares calculations, especially with reference to parallelism.

6. Acknowledgement. We thank our colleague Michael Todd for suggesting a number of changes that have improved the presentation.

REFERENCES

[1] R. H. BARTELS, A. R. CONN, AND J. W. SINCLAIR, *Minimization techniques for piecewise differentiable functions: the l_1 solution to an overdetermined linear system*, Siam J. Numer. Anal., 15 (1978), pp. 224–240.

[2] T. F. COLEMAN AND A. R. CONN, *Second-order conditions for an exact penalty function*, Mathematical Programming, 19 (1980), pp. 178–185.

[3] T. F. COLEMAN AND Y. LI, *A global and quadratic affine scaling method for (augmented) linear l_1 problems*, in Proceedings of the 1989 Dundee Conference on Numerical Analysis, 1989.

[4] ———, *A global and quadratic affine scaling method for linear l_1 problems*, Tech. Rep. 89–1026, Computer Science Department, Cornell University, 1989.

[5] M. MEKETON, *Least absolute value regression*, tech. rep., AT&T Bell Laboratories, Holmdel, 1988.

[6] C. B. MOLER, J. LITTLE, S. BANGERT, AND S. KLEIMAN, *ProMatlab User's guide*, MathWorks, Sherborn, MA, 1987.

[7] B. MURTAGH AND M. SAUNDERS, *MINOS 5.1 user's guide*, Tech. Rep. SOL-83-20R, Stanford University, 1987.

[8] C. VAN LOAN, *On the method of weighting for equality-constrained least squares problems*, SIAM Journal on Numerical Analysis, 22 (1985), pp. 851–864.

[9] R. J. VANDERBEI, M. S. MEKETON, AND B. A. FREEDMAN, *A modification of Karmarkar's linear programming algorithm*, Algorithmica, 1 (1986), pp. 395–407.