SIAM J. NUMER. ANAL.
Vol. 15, No. 2, April 1978

MINIMIZATION TECHNIQUES FOR PIECEWISE DIFFERENTIABLE FUNCTIONS: THE l_1 SOLUTION TO AN OVERDETERMINED LINEAR SYSTEM*

RICHARD H. BARTELS,† ANDREW R. CONN‡ AND JAMES W. SINCLAIR¶

Abstract. A new algorithm is presented for computing a vector x which satisfies a given m by n $(m>n\geqq 2)$ linear system in the sense that the l_1 norm is minimized. That is, if A is a matrix having m columns $a_1,\cdots,a_m$ each of length n, and b is a vector with components $\beta_1,\ldots,\beta_m$, then x is selected so that

$$\phi(x)=\|A^Tx-b\|_1=\sum_{i=1}^{m}|a_i^Tx-\beta_i|$$

is as small as possible. Such solutions are of interest for the "robust" fitting of a linear model to data.

The function ϕ is directly minimized in a finite number of steps using techniques borrowed from Conn's approach toward minimizing piecewise differentiable functions. In these techniques if x is any point and $A_{\mathscr{Z}}$ stands for the submatrix consisting of those columns a_j from A for which the corresponding residuals $a_j^Tx-\beta_j$ are zero, then the discontinuities in the gradient of ϕ at x are handled by making use of the projector onto the null space of $A_{\mathscr{Z}}^T$.

Attention has been paid both to numerical stability and efficiency in maintaining and updating a factorization of $A_{\mathscr{Z}}$ from which the necessary projector is obtainable.

The algorithm compares favorably with the best so far reported for the linear l_1 problem, and it can easily be extended to handle linear constraints.

1. Introduction, statement of the problem, notation and general remarks. For the sake of notational unity we wish to consider all vectors to be column vectors. Consequently we will state the problem in a form which is at slight variance with common usage.

Let A be an n by m $(m>n\geqq 2)$ real matrix with columns $a_1,\ldots,a_m$. Let b be a vector of m real components $\beta_1,\ldots,\beta_m$. We wish to solve the overdetermined system of linear equations

$$A^Tx=b \tag{1.1}$$

in the l_1 sense. That is, we wish to find a real vector x having n components which solves the problem

$$\text{minimize } \phi(x)=\|A^Tx-b\|_1=\sum_{i=1}^{m}|a_i^Tx-\beta_i|. \tag{1.2}$$

For the moment no constraints are imposed on x. Linear inequality constraints are, however, easily accommodated into the problem, as will be mentioned in a later section.

For any point x the index set

$$\mathscr{Z}=\{i|a_i^Tx-\beta_i=0\}=\{i_1,\ldots,i_k\} \tag{1.3}$$

* Received by the editors January 28, 1976, and in revised form April 15, 1977. This research was supported in part by the National Science Foundation of the United States Government under Grant DCR75-07817 and in part by the National Research Council of the Canadian Government under Grant A8639.

† Mathematical Sciences Department, Johns Hopkins University, Baltimore, Maryland 21218.

‡ Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

¶ Mathematics Department, University of Dundee, Dundee, Scotland, Great Britain.

associated with satisfied equations and its complement $\mathscr{Z}^c$ will be of interest. Corresponding to this set we will have the matrix

$$A_{\mathscr{Z}} = [a_{i_1}, \ldots, a_{i_k}], \qquad i_j \in \mathscr{Z}, \tag{1.4}$$

consisting of the columns of A associated with the satisfied equations, and we will have the nullspace

$$\mathscr{N} = \mathscr{N}(A_{\mathscr{Z}}^T) = \{y | a_i^T y = 0 \text{ for all } i \in \mathscr{Z}\}. \tag{1.5}$$

The orthogonal projector onto $\mathscr{N}$ will be denoted by $P_{\mathscr{N}}$.

At each point x under consideration the vectors

$$r = A^T x - b = [\rho_1, \ldots, \rho_m]^T \tag{1.6}$$

and

$$s = \operatorname{sgn}(r) = [\operatorname{sgn}(\rho_1), \ldots, \operatorname{sgn}(\rho_m)]^T = [\sigma_1, \ldots, \sigma_m]^T \tag{1.7}$$

will be referred to frequently.

Finally we will want to consider the set

$$\mathscr{A} = \{\alpha_l | l \in \mathscr{Z}^c \text{ and } \alpha_l = -\rho_l / a_l^T p \text{ and } \alpha_l > 0\} \tag{1.8}$$

for a chosen vector p.

We will be proposing an algorithm for problem (1.2) in the ensuing sections. It must be anticipated, as with many algorithms, that we will have to consider ours from two distinct points of view: the mathematical and the computational. From neither point of view will it be required that A have full rank. Linear dependencies in the matrix A associated with certain points x will, however, require special consideration from the mathematical standpoint but may be brushed somewhat aside from the computational standpoint. The dependencies which concern us correspond in concept to degenerate vertices in linear programming problems. There is reason for saying, as is done explicitly or implicitly with regard to the general linear programming problem, that degeneracy cannot exist in a world full of round-off error. We will subscribe to this philosophy in implementing our algorithm. But, as we will see in the next section, problem (1.2) is precisely equivalent to the linear programming problem. As the mathematical point of view demands some consideration of degeneracy when the linear programming problem is being handled, so must degeneracy be at least a passing issue for us in the l_1 problem.

2. The l_1 problem and linear programming. A number of authors have noted that problem (1.2) can be expressed as a linear programming problem, and this has become the basis for some of the best l_1 algorithms. The paper by Barrodale and Roberts [1] is to be recommended for its discussion and bibliography in this regard.

What is less often pointed out is that the linear programming problem, in turn, can be expressed as an l_1 problem. We follow the development of Witzgall [15] below to establish the equivalence of the two problems.

We may reformulate (1.2) as

$$\begin{aligned} &\text{minimize } e^T w = \sum_{i=1}^{m} \omega_i \\ &\text{subject to } w_i \geqq |a_i^T x - \beta_i| \quad \text{for } i = 1, \ldots, m, \end{aligned} \tag{2.1}$$

where $e = [1, \ldots, 1]^T$ and $w = [\omega_1, \ldots, \omega_m]^T$.

This is equivalent to

$$\begin{aligned} &\text{minimize } e^T w \\ &\text{subject to } \omega_i \geqq (a_i^T x - \beta_i) \\ &\quad \text{and } \omega_i \geqq (\beta_i - a_i^T x) \quad \text{for } i = 1, \ldots, m. \end{aligned} \tag{2.2}$$

In matrix-vector notation this becomes

$$\begin{aligned} &\text{minimize } [e^T, 0^T]\begin{bmatrix} w \\ x \end{bmatrix} \\ &\text{subject to } \begin{bmatrix} I & -A^T \\ I & +A^T \end{bmatrix}\begin{bmatrix} w \\ x \end{bmatrix} \geqq \begin{bmatrix} -b \\ +b \end{bmatrix}. \end{aligned} \tag{2.3}$$

The dual of (2.3) is the linear programming problem

$$\begin{aligned} &\text{maximize } -u^T b + v^T b \\ &\text{subject to } \quad u + v = e, \\ &\qquad\qquad Av - Au = 0, \\ &\qquad\qquad u \geqq 0, \quad v \geqq 0. \end{aligned} \tag{2.4}$$

After appropriate substitutions, $v = e - u$ and $y = 2u - e$, we obtain

$$\begin{aligned} &\text{minimize } b^T y \\ &\text{subject to } Ay = 0 \\ &\quad \text{and } -e \leqq y \leqq +e. \end{aligned} \tag{2.5}$$

To move in the other direction, we note that any bounded, feasible linear programming problem can be expressed in the form

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax = b \\ &\quad \text{and } -h \leqq x \leqq +h \end{aligned} \tag{2.6}$$

for some (suitably large) bound h on the variables.

An appropriate scaling of the variables will change the last condition to $-e \leqq x \leqq +e$. Further, if μ is a suitably large positive number and ζ is a variable, the following is equivalent to (2.6), because ζ will be forced to have the value $+1$ at any optimum:

$$\begin{aligned} &\text{minimize } c^T x - \mu\zeta \\ &\text{subject to } Ax - \zeta b = 0, \quad -e \leqq x \leqq +e, \quad -1 \leqq \zeta \leqq 1. \end{aligned} \tag{2.7}$$

This, in turn, is simply a problem of the form (2.5), which makes it the dual of an l_1 problem.

This establishes the equivalence. We can expect no shortcuts in studying the l_1 problem.

In a later section we wish to present some comparisons of our method with that given by Barrodale and Roberts in [1], [2]. To do this we note that their formulation of

(1.2) as a linear programming problem is the following:

$$\begin{aligned} &\text{minimize } e^T(u+v) \\ &\text{subject to } A^T(y-z)-(u-v)=b \\ &\text{and } y\geqq 0,\quad z\geqq 0,\quad u\geqq 0,\quad v\geqq 0, \end{aligned} \tag{2.8}$$

where x has been split into a positive part y and a negative part z, and the residual vector A^Tx-b has similarly been split into positive and negative components u and v. In a more complete matrix formulation (2.8) can be written as

$$\begin{aligned} &\text{minimize } [e^T, e^T, 0^T, 0^T]\begin{bmatrix} u \\ v \\ y \\ z \end{bmatrix} \\ &\text{subject to } [-I, +I, +A^T, -A^T]\begin{bmatrix} u \\ v \\ y \\ z \end{bmatrix}=b \\ &\text{and } [u^T, v^T, y^T, z^T]\geqq 0^T. \end{aligned} \tag{2.9}$$

Barrodale and Roberts use a variant of the simplex method to solve (2.9) which takes account of the special structure of this problem and the special relationship between the components of y and those of z.

Our own approach to solving (1.2) is to regard it as an exercise in the unconstrained minimization of a simple, piecewise differentiable, function. Our predecessors in this direct approach are Claerbout and Muir [4], who have proposed a somewhat more rudimentary algorithm than ours which does not involve an explicit use of projectors.

3. Linear changes in the argument of ϕ. Let x be any point, and regard the n-component vector p as defining a direction of motion away from x. We are interested in determining the value of ϕ at a translated point $x+\alpha p$ for $\alpha>0$.

$$\begin{aligned} \phi(x+\alpha p) &= \sum_{i=1}^{m} [a_i^T(x+\alpha p)-\beta_i]\,\operatorname{sgn}[a_i^T(x+\alpha p)-\beta_i] \\ &= \sum_{i\in\mathscr{Z}^c} [\rho_i+\alpha a_i^T p]\,\operatorname{sgn}[\rho_i+\alpha a_i^T p]+\alpha \sum_{i\in\mathscr{Z}} [a_i^T p]\,\operatorname{sgn}[a_i^T p], \end{aligned} \tag{3.1}$$

where we are using the fact that $\rho_i=a_i^Tx-\beta_i$ is zero whenever $i\in\mathscr{Z}$. If $\alpha>0$ is small enough, specifically if

$$0<\alpha<\min \mathscr{A}, \tag{3.2}$$

where $\mathscr{A}$ is as given in (1.8), then $\operatorname{sgn}[\rho_i+\alpha a_i^Tp]=\operatorname{sgn}[\rho_i]=\sigma_i$ for all $i\in\mathscr{Z}^c$. Therefore (3.1) becomes

$$\phi(x+\alpha p)=\sum_{i\in\mathscr{Z}^c}|\rho_i|+\alpha\Big(\sum_{i\in\mathscr{Z}^c}\sigma_i a_i^Tp+\sum_{i\in\mathscr{Z}}|a_i^Tp|\Big). \tag{3.3}$$

If we define a vector h by

$$h=\sum_{i\in\mathscr{Z}^c}\sigma_i a_i \tag{3.4}$$

then (3.3) takes on the character of a Taylor expansion:

$$\phi(x+\alpha p)=\phi(x)+\alpha h^T p+\alpha \sum_{i\in\mathscr{Z}} |a_i^T p|. \tag{3.5}$$

4. Projection. If we consider only those direction vectors p which lie in the nullspace of $A_{\mathscr{Z}}^T$, as defined in (1.4) and (1.5), then the last term of (3.5) will be zero:

$$\phi(x+\alpha p)=\phi(x)+\alpha h^T p \quad \text{for all } p\in\mathscr{N}. \tag{4.1}$$

We need only determine some $p \in \mathscr{N}$ such that $h^T p < 0$ to produce a translated point $x + \alpha p$ at which the value of ϕ is strictly less than $\phi(x)$. In fact α could be chosen equal to the upper bound in (3.2) to achieve the greatest reduction in the value of ϕ with respect to all $\alpha > 0$ for which (4.1) holds true. By breaking the bondage of this upper bound, as we shall see in a later section, even better choices of α can be made.

The easiest choice to make for p is

$$p=-P_{\mathscr{N}}h; \tag{4.2}$$

that is, the projection of $-h$ onto the nullspace of $A_{\mathscr{Z}}^T$, as long as this projection is nonzero. If this projection is zero, the corresponding point x will be called a *dead point.* Such points include the set of all optimal points for (1.2).

5. Optimality and dead points. Consider general direction vectors p; that is, consider (3.3) and (3.5) once again. Clearly $\phi(x+\alpha p)\geqq\phi(x)$ for all $\alpha>0$ close enough to zero if and only if

$$h^T p+\sum_{i\in\mathscr{Z}} |a_i^T p|\geqq 0 \quad \text{for all } p. \tag{5.1}$$

Should this be the case, then x provides a local minimum for the function ϕ. But $\phi(x)=\|A^T x-b\|_1$ is a convex function; hence local minima are global minima. Thus x is optimal for (1.2) if and only if (5.1) holds true.

The remark should be made that the solution to the l_1 problem, as in the case for the linear programming problem, need not be unique.

Suppose that $P_{\mathscr{N}}h$ is not zero. Then the vector p chosen as in (4.2) will violate (5.1); hence the corresponding x cannot be optimal.

On the other hand let us assume that $P_{\mathscr{N}}h = 0$; that is, the corresponding x is a dead point. Then h lies in the span of $A_{\mathscr{Z}}$:

$$h=A_{\mathscr{Z}}w=\sum_{j=1}^{k} \omega_j a_{i_j} \tag{5.2}$$

for some $w=[\omega_1,\ldots,\omega_k]^T$ and for $i_j\in\mathscr{Z}$. Consequently (5.1) can be rewritten in the form

$$\sum_{j=1}^{k} [1+\operatorname{sgn}(a_{i_j}^T p)\omega_j]|a_{i_j}^T p|. \tag{5.3}$$

Note that w is unique if and only if the columns of $A_{\mathscr{Z}}$ are linearly independent. A point x at which this is true is a *nondegenerate dead point.* Degenerate dead points will be considered in a separate section, so assume that w is unique.

Note that (5.3) can be made negative only if $|\omega_{j_0}|>1$ for some j_0. For if $|\omega_j|\leqq 1$ for all j, then all terms of the sum must be nonnegative no matter what p is considered.

Assume that a j_0 as above exists, and let

$$p=-\operatorname{sgn}(\omega_{i_0})P_{\mathscr{N}_{i_0}}a_{i_{i_0}}, \tag{5.4}$$

where $\mathcal{N}_{j_0}$ is the nullspace of $A_{\mathcal{Z}}$ with column $a_{i_{j_0}}$ deleted. For this choice of p we have $p^T a_i = 0$ if $i \in \mathcal{Z} - \{i_{j_0}\}$. And we have

$$\operatorname{sgn}(p^T a_{i_{j_0}})\omega_{j_0} = -|\omega_{j_0}|, \tag{5.5}$$

which gives a negative value to (5.3). Thus a nondegenerate dead point is optimal if and only if $|\omega_j| \leqq 1$ for all $j = 1, \ldots, k$ as given above.

6. The restricted gradient and the choice of α. Implicit in the foregoing is the result that either x is optimal, or else there is a p such that

$$\phi(x+\alpha p) = \phi(x) + \alpha p^T g \quad \text{with } p^T g < 0 \tag{6.1}$$

for all α satisfying (3.2). In case x is not a dead point, we may take

$$g = h \quad \text{and} \quad p = -P_{\mathcal{N}} h \tag{6.2}$$

under no assumptions about the rank of A. In case x is a nondegenerate dead point, we may take

$$g = h + \operatorname{sgn}(p^T a_{i_{j_0}}) a_{i_{j_0}} \quad \text{and} \quad p = -\operatorname{sgn}(\omega_{j_0}) P_{\mathcal{N}_{j_0}} a_{i_{j_0}}, \tag{6.3}$$

where j_0 is the index of a component of w, as given in (5.2), for which $|\omega_{j_0}| > 1$. We will refer to g as the *restricted gradient* of the function ϕ. When the restricted gradient of ϕ exists under the above rules, it is possible to produce a point $x + \alpha p$ at which the value of ϕ is lower than $\phi(x)$. Our assumptions would permit us to make α no larger than the bound given in (3.2). However, if α is made marginally greater than this bound, then it is easily established that the restricted gradient at the point $x + \alpha p$ for such a value of α is

$$g' = g - 2 \sum_{l \in \mathcal{L}} \sigma_l a_l \tag{6.4}$$

where $\mathcal{L}$ is that set of indices from $\mathcal{Z}^c$ on which the upper bound of (3.2) is taken. In nondegenerate problems, in fact, $\mathcal{L}$ will consist of a single index l' so that

$$g' = g - 2\sigma_{l'} a_{l'} \tag{6.5}$$

where l' gives the minimum of the positive ratios described in (1.8).

It may still be true that $p^T g' < 0$; that is

$$p^T g < 2p^T \Big(\sum_{l \in \mathcal{L}} \sigma_l a_l \Big), \tag{6.6}$$

in which case α may be increased to the next larger value in the set $\mathcal{A}$ for a further decrease in ϕ. We may continue running past values in the set $\mathcal{A}$, adjusting the restricted gradient according to (6.4) and applying the test in (6.6).

This process must, however, terminate at some value $\alpha \in \mathcal{A}$, for if we could increase α without bound, ϕ could be decreased without bound, which is not the case. When a maximal $\alpha \in \mathcal{A}$ has been found, we may reset x to $x + \alpha p$ and reconsider the question of finding a descent direction for ϕ from this new point.

7. The algorithm—Simple, nondegenerate case.

(0) Select any point x.

(1) (i) Identify $\mathcal{Z} = \{i_1, \ldots, i_k\}$.
Let $A_{\mathcal{Z}}$ and $\mathcal{N}$ be as in (1.4) and (1.5) respectively.

(ii) Compute h according to (3.4).

(iii) Compute $p = -P_{\mathcal{N}} h$.

If $p \neq 0$, let $g = h$, and go to (2).

(iv) Compute w according to (5.2).

(v) If $|\omega_j| \leqq 1$ for all $j = 1, \ldots, k$ stop.
In this case x is optimal.

(vi) Find $i_{j_0} \in \mathscr{Z}$ such that $|\omega_{i_0}| > 1$.

(vii) Change $\mathscr{Z}$ to $\mathscr{Z} - \{i_{j_0}\}$, and make corresponding changes to $A_{\mathscr{Z}}$ and $\mathscr{N}$. Compute

$$p = -\operatorname{sgn}(\omega_{i_0}) P_{\mathscr{N}} a_{i_{j_0}}$$

and let

$$g = h - \operatorname{sgn}(\omega_{i_0}) a_{i_{j_0}}.$$

(2) Determine the elements of $\mathscr{A}$ as given in (1.8) and order them:

$$0 < \alpha_{l_1} < \alpha_{l_2} < \cdots < \alpha_{l_\nu}$$

where α_{l_j} cporresponds to $-\rho_{l_j}/a_{l_j}^T p$. Let $\nu = 1$.

(3) If $p^T g \geqq 2\sigma_{l_\nu} p^T a_{l_\nu}$, then go to (5).

(4) Change g to $g - 2\sigma_{l_\nu} a_{l_\nu}$.
Change ν to $\nu + 1$.
Go to (3).

(5) Replace x by $x + \alpha_{l_\nu} p$, and go to (1).

(Note that our action upon reaching a dead point is that suggested by the discussion of § 5. This is convenient, but it may not be optimal, a matter which will be alluded to from time to time.)

8. Finite step convergence. Starting at any point, the algorithm generates a sequence of further points, each member x' of the sequence being obtained from its predecessor x at step (5). The parameter α_{l_ν} in step (5) has been chosen so that the new point satisfies, among others, that equation associated with the index l_ν of step (3). If the point x' is not a dead point, then the direction vector p for generating x'', the succeeding point, will be chosen at step (1) (iii). This will guarantee that x'' satisfies all of the equations satisfied by x'. Step (5), once again, will ensure that x'' satisfies at least one additional equation of its own. Furthermore, p will have been chosen so that $\phi(x'') < \phi(x')$.

Thus, unless a dead point is encountered, the set of equations satisfied by each new point strictly contains the set of equations satisfied by its predecessor, and a strictly decreasing sequence of values is produced from ϕ.

One consequence of the above is the fact that steps (1) through (5) cannot be executed more than m times in succession before a dead point $\hat{x}$ is reached.

For dead points the determination of whether a vector p exists which will serve as a descent direction is given in steps (1) (iv) through (1) (vii).

We have only included the means to treat nondegenerate dead points in step (1) of the algorithm; because as we will contend in § 10, most problems may be regarded as nondegenerate for the purpose of a computer implementation.

Observe that a submatrix $\hat{A}_{\mathscr{Z}}$ of A is associated with the dead point $\hat{x}$. Correspondingly associated with $\hat{x}$ is the subspace $\mathscr{N}(\hat{A}_{\mathscr{Z}}^T)$. Note that (4.1) and (5.2) imply that ϕ is constant throughout this nullspace, so we may consider this constant value $\hat{\phi} = \phi(\hat{x})$ as characteristic of that dead point, of its associated submatrix $\hat{A}_{\mathscr{Z}}$ and of the subspace $\mathscr{N}(\hat{A}_{\mathscr{Z}}^T)$.

Suppose a vector p is found at step (1) for which (5.1) is violated. Then $\phi(\hat{x} + \alpha p) < \hat{\phi}$ for some $\alpha > 0$, and for all points which the algorithm generates from here on, the

value of ϕ will stay strictly less than $\hat{\phi}$. This means that no dead point associated with $\hat{A}_{\mathscr{X}}$ can ever be encountered after $\hat{x}$ is left behind.

Hence we may conclude that only a finite number of executions of steps (1) through (5) can occur between any two dead points, and that only a finite number of dead points can be generated, since each is associated with a submatrix of A which we must encounter only once. On the other hand, some dead point is optimal, and the algorithm will not terminate until an optimal dead point has been found.

9. The algorithm—Degeneracy. Steps 1 (iv)-1 (vii) of the algorithm given in § 7 become invalid in the degenerate case, as does the assumption of a strict ordering for the α's in step (2).

The latter problem is easily accommodated by the following changes to steps (2)–(4):

(2) Determine the elements of $\mathscr{A}$ as given in (1.8) and order them

$$0<\alpha_{l_1}\leqq\alpha_{l_1}\leqq\cdots\leqq\alpha_{l_t}.$$

Set $\nu=1$.

(3) Let $\mathscr{L}_\nu=\{l|\alpha_l=\alpha_{l_\nu}\}$.
If $p^Tg\geqq 2p^T\sum_{l\in\mathscr{L}_\nu}\sigma_l\alpha_l$, then go to (5).

(4) Change g to $g-2\sum_{l\in\mathscr{L}_\nu}\sigma_l\alpha_l$.
Change ν to $\nu+\lambda_\nu$, where λ_ν is the number of elements in $\mathscr{L}_\nu$.

For the impact of degeneracy on (1) (iv)–(vii) we must return to (5.2). The vector w which expresses h as a linear combination of the columns of $A_{\mathscr{X}}$ is now no longer unique. However we can write w as

$$w=A_{\mathscr{X}}^{+}h+v=\hat{w}+v, \tag{9.1}$$

where v is any element of $\mathscr{N}(A_{\mathscr{X}}^T)$ and $A_{\mathscr{X}}^{+}$ is the (Moore–Penrose) generalized inverse of $A_{\mathscr{X}}$. This means that (5.3) can be rewritten as

$$\begin{aligned}&\sum_{j=1}^{k}[(\hat{w}_j+v_j)a_{i_j}^Tp+|a_{i_j}^Tp|]\\&=\sum_{j=1}^{k}[|a_{i_j}^Tp|+\hat{w}_ja_{i_j}^Tp]+\sum_{j=1}^{k}v_jp^Ta_{i_j}\\&=\sum_{j=1}^{k}[1+\hat{w}_j\operatorname{sgn}(a_{i_j}^Tp)]|a_{i_j}^Tp|+p^TA_{\mathscr{X}}v.\end{aligned} \tag{9.2}$$

But, since $p^TA_{\mathscr{X}}v=0$, the last expression reduces to (5.3) with $\hat{w}=A_{\mathscr{X}}^{+}h=[\hat{w}_1,\cdots,\hat{w}_k]^T$ replacing $w=[w_1,\cdots,w_k]^T$.

The linear dependence of the columns of $A_{\mathscr{X}}$ is bothersome in another regard. It is now no longer necessarily true that indices j_0 and vectors p can be chosen so that $a_{i_{j_0}}^Tp\neq 0$ while $a_{i_j}^Tp=0$ for $j\neq j_0$. This will be true in general only when the columns of $A_{\mathscr{X}}$ are independent. However, if we express $A_{\mathscr{X}}$ as QRV^T for matrices V and Q having orthonormal columns and R being nonsingular, which is possible unless $A_{\mathscr{X}}=0$ (a case we rule out) then we can express a_{i_j} as

$$a_{i_j}=Qs_j=Q(RV^Te_j)\qquad(j=1,\cdots,k) \tag{9.3}$$

where e_j is the jth coordinate vector. On the other hand,

$$A_{\mathscr{X}}^{+}=VR^{-1}Q^T. \tag{9.4}$$

Formulas (9.1)–(9.4) can now be combined to yield the result that an optimal point has

been found at step (1) of the algorithm if and only if no p can be selected such that

$$\sum_{j=1}^{k} [1+\hat{\omega}_j \operatorname{sgn}(s_j^T Q^T p)]|s_j^T Q^T p| < 0. \tag{9.5}$$

But $p = Qt + y$ for arbitrary $y \in \operatorname{span}(Q)^{\perp}$. Thus steps (1) (iv)–(vii) reduce to the problem of finding a vector t such that

$$\sum_{j=1}^{k} [1+\hat{\omega}_j \operatorname{sgn}(s_j^T t)]|s_j^T t| < 0. \tag{9.6}$$

Clearly the scaling of t is not important; we may restrict our attention to vectors of unit length (in any convenient norm). We are, in fact, left with the problem of exhausting the various combinations of $s_j/\|s_j\|$ to see which, if any, will satisfy (9.6), and reconstructing p from Q for the purposes of steps (2)–(5) in case our testing proves successful.

We should note that the whole of (1) even in this degenerate case requires no more than a finite number of steps to be carried out. The argument of finite convergence, on the other hand, is not "sensitive to the details of (1)." The argument still applies. It is important only that we step away from each nonoptimal dead point in a manner which strictly decreases the function ϕ, and that we spend no more than a finite number of steps getting from one dead point to the next.

We have included the above section chiefly to provide an insight into the mathematical difficulties which degeneracies pose. Computationally the problem can be treated more easily, as we propose below.

10. Implementation—Degeneracy. One should not lose sight of the fact that any algorithm will have to be executed on a computer for all but trivial problems. Often the computer imposes such restrictions on time, storage or accuracy that a particular algorithm must be rejected out of hand. On the other hand a computer's limitations can occasionally be used to advantage in the implementation of an algorithm. We believe that the resolution of degeneracies is a case in point.

The word length of a computer imposes a limitation on the accuracy with which a given problem can be stored in memory. If a floating-point number is composed of t digits regarded as a base β number, then any two given problems whose corresponding data elements differ from each other by one part in β^t or less will be regarded as identical by that computer. If neither of those data sets corresponds to an l_1 problem with a degenerate dead point while their common representation inside the machine is degenerate, then any difficulties which an algorithm might encounter on that data in that computer are merely unreasonable artifacts of the machine.

Since the data has been perturbed, possibly into degeneracy, by the act of storing it in the computer, we have chosen to regard all degeneracies as artifacts of the computer's limited word length. By reducing the accuracy of the machine slightly more, we will resolve these degeneracies in the same way they arise. We have tested an implementation of our algorithm which regards numbers differing by anything less than one part in $\beta^{t-\mu}$, where μ is a selected small integer, as identical numbers. In effect our program reserves the right to diminish the computer's accuracy by μ significant digits in the base β of its arithmetic.

This restriction is not normally invoked. But should a degeneracy be detected; that is, should the columns of $A_{\mathcal{Z}}$ prove to be linearly dependent to within a tolerance near one part in β^t, random perturbations of one half part in $\beta^{t-\mu}$ are added to one or more of the columns of $A_{\mathcal{Z}}$ until linear independence within the tolerance has been attained. These perturbations are undetectable at the accuracy of $t-\mu$ significant figures. We are

still solving the given problem to within the stated accuracy (one part in $\beta^{t-\mu}$), but we are now treating an incarnation of the problem in which degeneracy has been resolved. This scheme has proved entirely satisfactory on a wide variety of problems.

The above proposal could be made for the simplex algorithm as well. There it would be patent nonsense for certain classes of "precise" problems—those which can be expressed and manipulated exactly within the computer—such as, for example, transportation problems. Such problems are solved by special methods, and corresponding classes of l_1 problems should have their corresponding special methods. We feel that our perturbation techniques, however, are suitable for general "imprecise" l_1 problems—for example those arising from data-fitting.

11. Implementation—Projectors. If the n by k matrix $A_{\mathscr{X}}$ has independent columns, it can be factored into the form

$$A_{\mathscr{X}} = Q\begin{bmatrix} R \\ \hline 0 \end{bmatrix} = [Q_1 \vdots Q_2]\begin{bmatrix} R \\ \hline 0 \end{bmatrix} \tag{11.1}$$

where $Q = [Q_1 \vdots Q_2]$ is an n by n orthogonal matrix and R is k by k, right triangular, and nonsingular. Q_1 denotes the first k columns of Q, and Q_2 denotes the remaining $n-k$ columns.

The projector on $\mathscr{N}(A_{\mathscr{X}}^T)$ is given by the matrix

$$Q_2 Q_2^T, \tag{11.2}$$

and the vector w satisfying (5.2) is given by

$$w = R^{-1} Q_1^T h. \tag{11.3}$$

During the course of the algorithm we have to be concerned with updating (11.1) when a column is added to or deleted from $A_{\mathscr{X}}$. This task arises with such frequency that a wealth of techniques have been developed to handle equations (11.1)–(11.3) or equivalent formulas under just such changes to $A_{\mathscr{X}}$. Methods have been presented for keeping Q_1 and R alone, for keeping Q_2 and R alone, for dispensing with Q entirely, for maintaining Q as WD using some diagonal matrix D, for maintaining WD in product form, for expressing R as $\bar{D}U$, where U is unit triangular and $\bar{D}$ is diagonal, and so forth. It is not our place to review this material. References [6], [8], [9], [10], [11], [12], [14] will offer a sufficient background.

Our decision has been to maintain Q_1, Q_2 and R in the form

$$A_{\mathscr{X}} = WD\begin{bmatrix} U \\ \hline 0 \end{bmatrix} \tag{11.4}$$

where D is a diagonal matrix with positive diagonal entries, and $Q = WD^{1/2}$, $D^{1/2}U = R$. We have begun our investigations with problems small enough so that $A_{\mathscr{X}}$ fits entirely in computer storage, and we feel that motion away from dead points should not necessarily be restricted to the simplistic rule given in § 7, steps (1) (vi)–(vii). The former observation means that we may ignore product form representations for the present, leaving them to the later development of programs for large sparse problems. The latter observation forces us to maintain some generality in our matrix factorizations. We cannot yet predict ratios of the instances of column deletion to the instances of column addition during the course of the algorithm. Thus it is not clear that Q_1 or Q_2 should or should not be explicitly maintained.

Were we to commit ourselves to steps (1) (vi)–(vii) as given, then simple test problems have indicated that the vast majority of iteration cycles involve motion

directly from dead point to dead point. Each dead point encountered is characterized by n active equations. And the neighboring dead points are encountered via the exchange of one active for one inactive equation. This suggests that, were the simple algorithm optimal, one ought to choose the decomposition at variance with (11.4) to take advantage of this observed behavior.

The diagonal matrix D in (11.4) is a device for avoiding the computation of square roots in setting up and applying Givens transformations. We will follow the outline in [9] and ignore the matrix D for clarity of presentation.

If column j_0 is deleted from $A_{\mathcal{X}}$ the resulting matrix $\bar{A}_{\mathcal{X}}$ will have the factorization

$$\bar{A}_{\mathcal{X}} = Q\begin{bmatrix} H \\ \hline 0 \end{bmatrix}, \tag{11.5}$$

where H is right Hessenberg and is obtained from R by deleting its j_0th column. If k were 6 and j_0 were 2, for example, the matrix H would have the form:

$$\begin{bmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{bmatrix}. \tag{11.6}$$

The subdiagonal elements in H, which extend from the new j_0th column (the old (j_0+1)st column of R) to the last column can be removed by applying an appropriate sequence of transformations of the form

$$G_\nu = \begin{bmatrix} 1 & & & & 0 \\ & \ddots & & & \\ & & 1 & & \\ & & & \gamma & \delta \\ & & & \delta & -\gamma \\ & & & & 1 \\ 0 & & & & & \ddots & 1 \end{bmatrix}, \tag{11.7}$$

where γ and $-\gamma$ occupy positions ν and $\nu+1$ on the diagonal, and $\gamma^2+\delta^2=1$.

Thus (11.5) is to be transformed into

$$\bar{A}_{\mathcal{X}} = [QG_{j_0}\cdots G_{k-1}][G_{k-1}\cdots G_{j_0}]\begin{bmatrix} H \\ \hline 0 \end{bmatrix} = \bar{Q}\begin{bmatrix} \bar{R} \\ \hline 0 \end{bmatrix}. \tag{11.8}$$

To reverse the process, the addition of a column to $A_{\mathcal{X}}$ may be written as

$$\bar{A}_{\mathcal{X}} = [A_{\mathcal{X}}, a] = \left[Q\begin{bmatrix} R \\ \hline 0 \end{bmatrix}, a\right] = Q\left[\begin{array}{c|c} R & v_1 \\ \hline 0 & v_2 \end{array}\right] = [Q_1 \mid Q_2]\left[\begin{array}{c|c} R & v_1 \\ \hline 0 & v_2 \end{array}\right], \tag{11.9}$$

where $v_1 = Q_1^T a$ and $v_2 = Q_2^T a$. Once again a product of matrices of the form (11.7) can be used, this time to transform v_2 into a vector having zero in all components save the

first:

$$\bar{A}_{\mathscr{Z}} = [QG_{n-1}\cdots G_{k+1}][G_{k+1}\cdots G_{n-1}]\left[\begin{array}{c|c} R & v_1 \\ \hline 0 & v_2 \end{array}\right] = \bar{Q}\left[\begin{array}{c|c} R & v_1 \\ \hline 0 & d \end{array}\right] = \bar{Q}\left[\begin{array}{c} \bar{R} \\ \hline 0 \end{array}\right], \tag{11.10}$$

where $d = (\sqrt{v_2^T v_2}, 0, \cdots, 0)^T$.

12. Linear constraints. We begin by observing that only linear inequality constraints are of interest. Linear equality constraints can be handled exactly as in the l_2 case (e.g., see [14]). Briefly, suppose we must solve

$$\begin{aligned} &\text{minimize } \|A^T x - b\|_1 \\ &\text{subject to } F^T x = d. \end{aligned} \tag{12.1}$$

Then, assuming linear independence for the columns of F as well as for those of A, we factor F into

$$F = P\left[\begin{array}{c} S \\ \hline 0 \end{array}\right] \tag{12.2}$$

where P is orthogonal and S is right triangular and nonsingular. After this transformation the constraints become

$$F^T x = [S^T \mid 0](P^T x) = [S^T \mid 0]\left[\begin{array}{c} y \\ \hline v \end{array}\right] = d \tag{12.3}$$

where $P^T x = \left[\begin{array}{c} y \\ \hline v \end{array}\right]$. If λ denotes the number of columns of F, then y will have λ components and v will have $n-\lambda$. Clearly y is uniquely determined and v is free.

The minimization problem may now be written as

$$\begin{aligned} &\text{minimize } \|(A^T P)(P^T x) - b\|_1 \\ \equiv\ &\text{minimize } \left\| [E^T \mid C^T]\left[\begin{array}{c} y \\ \hline v \end{array}\right] - b \right\|_1 \\ \equiv\ &\text{minimize } \|C^T v - (b - E^T y)\|_1 \end{aligned} \tag{12.4}$$

which is simply an unconstrained l_1 problem with matrix C and right hand side $b - E^T y$. In this representation of the problem each of the columns of $P^T A$ has been partitioned into two sub-columns, an initial one of λ components and a final one of $n-\lambda$. The matrix E is the matrix whose columns are those initial segments, and C is the matrix whose columns are the final segments.

To handle an inequality constrained problem

$$\begin{aligned} &\text{minimize } \|A^T x - b\|_1 \\ &\text{subject to } F^T x \geqq d, \end{aligned} \tag{12.5}$$

where $F = [f_1, \cdots, f_\kappa]$ and $d = [\delta_1, \cdots, \delta_\kappa]^T$, we observe that the basic l_1 algorithm which we have presented is directly related to the linear programming algorithm of Conn [5]. In effect we are using an algorithm for minimizing the more general (penalty) function

$$\phi(x) = \mu\psi(x) - \sum_i \min(0, f_i^T x - \delta_i) + \sum_j |a_j^T x - \beta_j| \tag{12.6}$$

in the case where ψ is identically zero and the sum on i is vacuous.

As such, the addition of constraints presents no problems. We include them in our objective function as shown in (12.6). Cycle for cycle in the algorithm we need only recognize those constraints which are currently binding and include them in our projection; i.e., include them in $A_{\mathscr{Z}}$.

In the more general case of nonlinear constraints, or even the l_1 problem where $A^T x - b$ is replaced by a nonlinear function of x, we may use projections related to local linearizations of the nonlinear functions.

The above ideas will be developed in more detail in subsequent papers.

13. Further remarks. There are aspects of this algorithm which offer advantages even beyond the ease with which constraints can be accommodated. We mention two below.

I. The algorithm requires no transformations of the data matrix A. It has the feature in common with the nontableau implementations of the simplex method that the data can be left undisturbed at the cost of keeping a (relatively small) "basis matrix" on the side. Thus we feel that our method holds a significant advantage over many other l_1 algorithms in its applicability to large sparse structured problems.

II. Steps (1) (vi)–(vii), as we have formulated them, result in motion away from dead points which is associated in releasing the j_0th equation from the set $\mathscr{Z}$. This may not be the optimal descent direction to take away from a dead point. Some experiments have been conducted with modifications to (1) (vi)–(vii) which seek to release as many equations as possible from $\mathscr{Z}$ from among those associated with components ω_j of (1) (iv)–(v) having magnitude greater than one. Such techniques correspond to the use of "nonsimplex steps" or "non-vertex-vertex steps" in linear programming, and they may be able to reduce the algorithm's average problem-solving time.

The algorithm of § 7 is the simplest which can be derived from the approach to the l_1 problem which we have taken in this paper. It is reasonable to ask whether this algorithm is distinct from the one proposed by Barrodale and Roberts in [1]. We shall offer some evidence below to the effect that it is. But even if that were not the case, we would argue that our approach has merit for several reasons. It provides a description of the algorithm in a form which leads easily to numerically stable implementations such as the one we have sketched in § 11. It presents an alternative discussion of the material surrounding linear l_1 problems which does not make use of linear programming, which is more direct and which provides a more geometrical description of the algorithm. It outlines a reasonable starting ground from which the abovementioned generalizations and extensions can be readily approached.

That the algorithm given in § 7 and the one given by Barrodale and Roberts are not identical can be seen from the example:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}, \qquad b^T = [1, 1, 2, 3, 2]$$

which is presented in [1]. We will investigate the algorithm of § 7 and that given in [1] using the starting point $x_0 = [1, 0]^T$.

Note that the Barrodale–Roberts algorithm can accommodate any starting point by translating the unknowns x to $x - x_0$. This means that the right-hand side vector b^T will have to be set to $[0, 0, 1, 2, 1]$ in this example for the Barrodale–Roberts algorithm, and the solution which is obtained will have to be translated back by the amount x_0.

The tableaux which are produced by the Barrodale–Roberts algorithm are as shown in Tables 1–3 (using the notation of [1]).

TABLE 1

Basis	R	b_1	b_2
u_1	0	1	1
u_2	0	1	2
u_3	1	1	3
u_4	2	1	4
u_5	1	1	5
MC	4	5	15

TABLE 2

Basis	R	c_1	u_5
v_1	$\frac{1}{5}$	$\frac{4}{5}$	$\frac{1}{5}$
v_2	$\frac{2}{5}$	$\frac{3}{5}$	$\frac{2}{5}$
u_3	$\frac{2}{5}$	$-\frac{2}{5}$	$-\frac{3}{5}$
u_4	$\frac{6}{5}$	$-\frac{1}{5}$	$-\frac{4}{5}$
b_2	$\frac{1}{5}$	$-\frac{1}{5}$	$\frac{1}{5}$
MC	$\frac{11}{5}$	$\frac{4}{5}$	$-\frac{9}{5}$

TABLE 3

Basis	R	v_1	u_5
c_1	$\frac{1}{4}$	$\frac{5}{4}$	$\frac{1}{4}$
v_2	$\frac{1}{4}$	$-\frac{3}{4}$	$\frac{1}{4}$
u_3	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
u_4	$\frac{5}{4}$	$\frac{1}{4}$	$-\frac{3}{4}$
b_2	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
MC	2	−1	−2

giving rise to the solution

$$[-\tfrac{1}{4}, \tfrac{1}{4}]^T,$$

or in terms of the original problem,

$$x^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{4} \\ \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \end{bmatrix},$$

which interpolates the 1st and 5th equations with error of approximation 2.

In contrast, our own algorithm proceeds as sketched below:

$x_0 = [1, 0]^T$ (starting point).
$\mathscr{L} = \{1, 2\}$.
$h = [-3, -12]^T$.
$P_N h = 0$ (x_0 is a dead point).
$w = [3, -6]$, so choose $j_0 = 2$.
$g = [-2, -10]^T$.
$p = [-1, 1]^T$.
$\alpha_3 = \frac{1}{2}$, $\alpha_4 = \frac{2}{3}$, $\alpha_5 = \frac{1}{4}$.
α_5 is chosen.
New point $= x_0 + \alpha_5 p = [\frac{3}{4}, \frac{1}{4}]$ *optimal.*

Thus our method has taken one step ("pivot") to arrive at the optimum while the Barrodale–Roberts method has taken two.

The two methods must, in fairness, be described as very similar in practice, even if they are not identical. In the next section we offer the results of comparisons on some test problems. In all cases the algorithm of § 7 and that given in [1] take roughly an equal number of steps to arrive at a solution.

14. Computational experience. A number of example problems were run on the Honeywell 6050 at Waterloo University to compare the program by Barrodale and Roberts appearing in [2] with our own method as given in [3]. The most efficient version of the code in [2] was used, i.e., that in which double subscripting is avoided.

In each problem we investigated the behavior of our program using $x = 0$ as a starting point (which is the starting point the Barrodale and Roberts program normally uses).

The Honeywell 6050 runs in a time-sharing mode at Waterloo, and it proved difficult to get accurate timings for the programs. The times given below are only intended to give a verification that the two programs run at roughly equal speed.

Example 1. Approximate the values of $f(z) = e^{-z}(z)$ taken for $z := 0.0$ step 0.02 until 4.0 (i.e. on 201 points) by a polynomial of degree $n - 1$.

TABLE 4
Example 1.

Barrodale/Roberts (Number of pivot steps)	Bartels/Conn/ Sinclair (Number of steps)	$m = 201$ $n =$ (below)
6	7	2
5	5	3
9	7	4
13	12	5
15	21	6
19	15	7
17	17	8
0.22	0.37	Total time in Seconds

Example 2. Approximate the values of $f(z) = e^z$ taken for $z := 0.0$ step 0.1 until 2.0 (i.e., on 201 points) by a polynomial of degree $n - 1$. Table 5 is to be read as in the previous example.

TABLE 5
Example 2.

B/R	B/C/S	$m = 201$ $n =$ (below)
3	4	2
6	7	3
7	13	4
10	14	5
14	15	6
17	13	7
0.22	0.29	Time

Example 3. Approximate the values of $f(z) = \sqrt{z}$ taken for $z = 0.0$ step 0.02 until 1.0 by a cubic spline with knots 0.1, 0.2, 0.4 and 0.7 ($m = 51$, $n = 8$). Both programs

were started with the same initial guess (zero).

Barrodale/Roberts:	19 Pivot Steps
Bartels/Conn/Sinclair:	18 Steps

Example 4. Calculate the best l_1 solution to the following overdetermined system of linear equations.

$$\begin{aligned}
5x_1+3x_2+4x_3+12x_4+\ 4x_5&=7,\\
9x_1+7x_2+3x_3+19x_4+13x_5&=4,\\
6x_1+6x_2+0x_3+12x_4+12x_5&=2,\\
9x_1+9x_2+7x_3+25x_4+11x_5&=7,\\
3x_1+0x_2+1x_3+\ 4x_4+\ 2x_5&=7,\\
8x_1+1x_2+8x_3+17x_4+\ 1x_5&=7,\\
1x_1+9x_2+8x_3+18x_4+\ 2x_5&=3,\\
3x_1+1x_2+1x_3+\ 5x_4+\ 3x_5&=5,\\
0x_1+9x_2+3x_3+12x_4+\ 6x_5&=3
\end{aligned}$$

$(m=9, n=5)$.

Remarks. The entries of the 1st, 2nd and 3rd columns were chosen from a table of random integers. The 4th column is the sum of the previous 3 columns. The 5th column is the sum of the first 2 columns minus the 3rd. Thus the matrix of coefficients has rank 3. The right-hand side was also chosen from a table of random integers.

With a zero starting point:

Barrodale/Roberts:	2 Pivot Steps
Bartels/Conn/Sinclair:	6 Steps

In all cases the B/R and B/C/S programs gave the same answers.

And finally, to demonstrate the capability of handling constraints, we include the results obtained from a modified version of the program applied to a problem given us by Maurice Cox at the National Physical Laboratory in Teddington, England. It arose in constructing a fit to given data $b_1, \cdots, b_m$ using a linear combination of cubic B-splines:

$$s(z)=\sum_{i=1}^{n} \xi_i B_i(z),$$

where the coefficients $\xi_1, \cdots, \xi_n$, are the unknowns. The equations arise from the demand that the residuals $s(z_j)-b_j$ be minimized in the l_1 sense on a given mesh of points $z_1, \cdots, z_m$. The constraints come from the demand that the function $s(z)$ be convex.

Our method, modified as indicated in (12.6) to handle inequality constraints, took 17 steps to solve the problem

$$\begin{aligned}
&\text{minimize } \|A^T x-b\|_1\\
&\text{subject to } C^T x \geqq d\\
&\text{where } x=[\xi_1, \cdots, \xi_n]^T
\end{aligned}$$

for the data (A), (b), (C), (d) given us by Cox as listed below. The residual l_1 norm was 0.6206897, and the coefficients were

$$\begin{aligned}
\xi_1&=\ 0.88793103\text{E}-01, & \xi_2&=\ 0.40517241\text{E}-01,\\
\xi_3&=-0.86296897\text{E}-03, & \xi_4&=-0.86206897\text{E}-03,\\
\xi_5&=-0.86206897\text{E}-03, & \xi_6&=\ 0.40517241\text{E}-01,\\
\xi_7&=\ 0.88793103\text{E}-01,
\end{aligned}$$

Data matrix (A).

$$\begin{array}{ccccccc} 8 & 32 & 8 & 0 & 0 & 0 & 0 \\ 1 & 23 & 23 & 1 & 0 & 0 & 0 \\ 0 & 8 & 32 & 8 & 0 & 0 & 0 \\ 0 & 1 & 23 & 23 & 1 & 0 & 0 \\ 0 & 0 & 8 & 32 & 8 & 0 & 0 \\ 0 & 0 & 1 & 23 & 23 & 1 & 0 \\ 0 & 0 & 0 & 8 & 32 & 8 & 0 \\ 0 & 0 & 0 & 1 & 23 & 23 & 1 \\ 0 & 0 & 0 & 0 & 8 & 32 & 8 \end{array}$$

Constraint matrix (C).

$$\begin{array}{ccccccc} 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{array}$$

Equation right-hand side (b).

$$2, 1, 0, 0, 0, 0, 0, 1, 2$$

The components of the constraint right-hand side (d) were all zero.

The answers have been checked independently on a linear programming routine.

Acknowledgments. We wish to thank M. A. Saunders of the Systems Optimization Laboratory at Stanford University, Christoph Witzgall of the National Bureau of Standards, Ian Barrodale of the University of Victoria and Maurice Cox of the National Physical Laboratory for their helpful comments.

REFERENCES

[1] I. BARRODALE AND F. D. K. ROBERTS, *An improved algorithm for discrete l_1 linear approximation*, this Journal, 10 (1973), pp. 839–848.

[2] ———, *Algorithm 478, Solution of an overdetermined system of equations in the l_1 norm*, Comm. ACM, 17 (1974), pp. 319–320.

[3] RICHARD H. BARTELS, ANDREW R. CONN AND JAMES W. SINCLAIR, *A FORTRAN program for solving overdetermined systems of linear equations in the l_1 sense*, Tech. Rep. 236, Mathematical Sciences Dept., Johns Hopkins Univ., Baltimore, MD, 1976.

[4] JON F. CLAERBOUT AND FRANCIS MUIR, *Robust modeling with erratic data*, Geophysics, 38 (1973), pp. 826–844.

[5] ANDREW R. CONN, *Linear programming via a non-differentiable penalty function*, this Journal, 13 (1976), pp. 145–154.

[6] J. W. DANIEL, W. B. GRAGG, L. KAUFMAN AND G. W. STEWART, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comput., 30 (1976), pp. 772–795.

[7] GEORGE FORSYTHE AND CLEVE B. MOLER, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[8] W. MORVEN GENTELMAN, *Least squares computations by Givens transformations without square roots*, J. Inst. Maths Appl., 12 (1973), pp. 329–336.

[9] PHILIP E. GILL, GENE H. GOLUB, WALTER MURRAY AND MICHAEL A. SAUNDERS, *Methods for modifying matrix factorizations*, Math. Comput., 28 (1974), pp. 505–535.

[10] PHILIP E. GILL AND WALTER MURRAY, *Two methods for the solution of linearly constrained and unconstrained optimization*, Tech. Rep. NAC 25, National Physical Laboratory, Teddington, Middlesex, England, 1972.

[11] PHILIP E. GILL, WALTER MURRAY AND MICHAEL A. SAUNDERS, *Methods for computing and modifying the LDV factors of a matrix*, Math. Comput., 29, (1975), pp. 1051–1077.
[12] DONALD GOLDFARB, *Matrix factorization in optimization of nonlinear functions subject to linear constraints*, Math. Programming, 10 (1976), pp. 1–31.
[13] G. HADLEY, *Linear Programming*, Addison-Wesley, Reading, MA, 1962.
[14] CHARLES L. LAWSON AND RICHARD J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
[15] CHRISTOPH WITZGALL, *On discrete l_1 approximation*, working paper.