SIAM J. NUMER. ANAL.
Vol. 10, No. 5, October 1973

AN IMPROVED ALGORITHM FOR DISCRETE l_1 LINEAR APPROXIMATION*

I. BARRODALE† AND F. D. K. ROBERTS‡

Abstract. By modifying the simplex method of linear programming, we are able to present an algorithm for l_1-approximation which appears to be superior computationally to any other known algorithm for this problem.

1. Introduction. This paper describes an algorithm which appears to be the most efficient yet devised for solving the general l_1-linear approximation problem. The algorithm is a modification of the simplex method applied to the primal formulation of the l_1-problem as a linear program.

The general l_1-linear approximation problem can be stated as follows. Let $f(x)$ be a given real-valued function defined on a discrete subset $X = \{x_1, x_2, \cdots, x_m\}$ of Euclidean space E^N. Given $n(\leqq m)$ real-valued functions $\varphi_j(x)$ defined on X, we form a linear approximating function $L(A, x) = \sum_{j=1}^{n} a_j \varphi_j(x)$ for any set $A = \{a_1, a_2, \cdots, a_n\}$ of real numbers. The l_1-problem is to determine a best approximation $L(A^*, x)$ which minimizes

$$\sum_{i=1}^{m} |f(x_i) - L(A, x_i)|. \tag{1}$$

It is well known that at least one best approximation always exists, and there are now several algorithms for calculating a best approximation. However, some of these algorithms require that the set of given functions $\{\varphi_1(x), \varphi_2(x), \cdots, \varphi_n(x)\}$ be linearly independent on X, or that it satisfy the Haar condition on X. This latter condition, which is equivalent to requiring that the given functions be linearly independent on *every* subset of n distinct elements from X, can be quite restrictive in practice.

As yet, it seems that the only use made of "least-first-power" approximations has been in the analysis of experimental data. Even in this situation, many scientists are unwilling to use estimates of parameters computed in a norm for which few statistical tests are available. However, there is no doubt that best l_1-approximations are often superior to best l_2-approximations when estimating the true form of data that contain some very inaccurate observations (see Rice and White [14], Barrodale [2], and Ekblom and Henriksson [10]).

The approximating functions used by experimental scientists for interpreting data are almost never polynomials, and they rarely constitute Haar sets (indeed, continuous functions of more than one independent variable *never* form a Haar set).

* Received by the editors September 29, 1971, and in revised form August 28, 1972. This work was supported in part by the National Research Council under Grants A5251 and A7143.

† Department of Mathematics, University of Victoria, Victoria, British Columbia, Canada. The work of this author, while visiting the Mathematics Research Center, Madison, Wisconsin, was sponsored by the United States Army under Contract DA-31-124-ARO-D-462.

‡ Department of Mathematics, University of Victoria, Victoria, British Columbia, Canada.

If the approximation problem is presented as the equivalent problem of solving an overdetermined system of linear equations, it may be quite difficult to decide beforehand on the rank of the given matrix.

Consequently, we feel that a data-fitting algorithm should impose as few restrictions as possible on the user's choice of approximating function. Since our algorithm is based on the simplex method it can be used with *any* linear approximating function.

The connection between linear programming and l_1-approximation was pointed out by Wagner [19] in 1959, although other more direct (but cumbersome) algorithms for the l_1-problem had been proposed much earlier (see Barrodale [2] for references to these).

In 1966 Barrodale and Young [8] described a primal algorithm which takes advantage of the special structure of the linear programming formulation of the l_1-problem. *The primal algorithm of the present paper is an improved version of this earlier simplex algorithm; in short, we have been able to reduce significantly the total number of iterations required, by discovering how to pass through several neighboring simplex vertices in a single iteration.*

In 1967 Usow [17] presented a descent method which attempts to calculate a best l_1-approximation by locating the lowest vertex of a convex polytope representing the set of all possible l_1-approximations. This iterative algorithm requires that the approximating function be formed from a Haar set, and it also stops prematurely if a certain degenerate situation arises. Usow [18] states that he has now corrected this latter deficiency and extended his algorithm to approximating functions formed from linearly independent sets of given functions.

In 1969 Robers and Ben-Israel [15] applied a new method for linear programming to the dual formulation of the l_1-problem. Their new method (which they call interval linear programming) is capable of solving any bounded-variable linear programming problem, and so it is natural to apply it to the l_1-problem in particular. Robers and Robers [16] have now supplied a special version of the general method of [15] which is designed specifically for the l_1-problem.

Finally, early in 1971 Abdelmalek [1] described an algorithm which determines best l_1-approximations as the limit of best l_p-approximations as $p \to 1^+$. His technique thus obtains a solution to a linear problem by solving a sequence of nonlinear problems.

Clearly, a comparison of the computational behavior of these various algorithms is both desirable and overdue. Consequently, we have conducted some numerical tests on (i) the latest descent method of Usow [18], (ii) the special adaptation by Robers and Robers [16] of the more general method of Robers and Ben-Israel [15], (iii) the bounded-variable simplex method of Dantzig [9] applied to the dual formulation of the l_1-problem, and (iv) our modified standard form of the simplex method applied to the l_1-problem in its primal form. (The numerical results given in Abdelmalek [1] demonstrate that his algorithm cannot possibly be as efficient as those selected above.)

Complete details of these test problems, together with the numerical results obtained and the central processor times involved, are contained in Barrodale and Roberts [5]. These comparisons leave no doubt as to the superiority of our new method (iv). We have subsequently made some coding improvements in the

FORTRAN program used to implement our method for this empirical study, and the latest version of this code is supplied in Barrodale and Roberts [6].

The l_1-problem (1) is restated as a linear programming problem in § 2, and our new algorithm is described in § 3.

2. Linear programming and l_1-approximation. For the l_1-problem (1) let us write $\varphi_{j,i} \equiv \varphi_j(x_i)$, $f_i \equiv f(x_i)$, and define nonnegative variables u_i, v_i, b_j, c_j by putting

$$f_i - \sum_{j=1}^{n} a_j \varphi_{j,i} = u_i - v_i, \qquad i = 1, 2, \cdots, m,$$

and $a_j = b_j - c_j$ for $j = 1, 2, \cdots, n$. Then a best l_1-approximation corresponds to an optimal solution to the (primal) linear programming problem:

$$\begin{aligned} &\text{Minimize} \quad \sum_{i=1}^{m} (u_i + v_i) \\ &\text{subject to} \quad f_i = \sum_{j=1}^{n} (b_j - c_j)\varphi_{j,i} + u_i - v_i, \quad i = 1, 2, \cdots, m, \\ &\text{and} \quad b_j, c_j, u_i, v_i \geqq 0. \end{aligned} \tag{2}$$

The formulation (2) appears in Barrodale and Roberts [4] along with the following result.

THEOREM. *If the column rank of the $m \times n$ matrix $\Phi = \{\varphi_{j,i}\}^T$ is $k(\leqq n)$, then there exists a best l_1-approximation which interpolates $f(x)$ in at least k points of X.*

Several authors (Wagner [19], Rabinowitz [13], Barrodale [3], for example) have suggested that the dual of (2) should be solved instead when m is large. The dual of (2) is stated most conveniently (e.g., Rabinowitz [13]) as the following bounded-variable linear programming problem:

$$\begin{aligned} &\text{Maximize} \quad \sum_{i=1}^{m} (w_i f_i - f_i) \\ &\text{subject to} \quad \sum_{i=1}^{m} w_i \varphi_{j,i} = \sum_{i=1}^{m} \varphi_{j,i}, \qquad j = 1, 2, \cdots, n, \\ &\text{and} \quad 0 \leqq w_i \leqq 2. \end{aligned} \tag{3}$$

In fact, as is confirmed by the results of our empirical study in [5], applying the bounded-variable simplex method of Dantzig [9] to (3) leads to a *less* efficient algorithm in general than solving the primal problem (2) by our version of the standard form of the simplex method.

There are, of course, several alternatives to the standard form of the simplex method which can be used to solve a linear programming problem (two forms of the revised simplex method, the primal-dual algorithm, the dual simplex algorithm, etc.). However, the denseness of the condensed tableau corresponding to (2), the availability of an initial basic feasible solution, and the simplicity with which we can implement our new idea of passing through several vertices in a single iteration, combine together to make the standard form of the simplex method the most economical algorithm for the l_1-problem.

3. The algorithm. A direct application of the simplex method to (2) does not yield an efficient algorithm. (On the other hand, this might well be a reasonable strategy if a particular problem arises involving additional (inequality) constraints on the approximating function.) Barrodale and Young [8] observed that an initial basic feasible solution to (2) is immediately available, and that most of the column vectors in the simplex tableau need not be stored explicitly. Barrodale and Roberts [4] recommended that for the first n iterations the choice of pivotal column be restricted to the columns associated with the variables b_j and c_j, and that thereafter a special pivotal column selection rule be employed. In spite of these improvements, the simplex method can take a large number of iterations to solve some problems. The reason for this can best be explained by considering the following particular problem.

Suppose that we wish to approximate $f(x) = e^x$ on 201 uniformly spaced points in the interval $[0, 2]$ by a straight line $L(A, x) = a_1 + a_2 x$. The initial simplex basis is provided by the column vectors associated with $u_1, u_2, \cdots, u_{201}$. If the simplex method with the above modifications is applied to this problem, the first iteration brings the vector associated with b_1 into the basis and forces the vector associated with u_1 out. The second iteration brings the vector corresponding to b_2 into the basis, and that corresponding to u_2 goes out. Thus, after two iterations, the simplex tableau represents an approximation which interpolates the first and second data points (since the current values of u_1, v_1, u_2, and v_2 are all zero).

Subsequent iterations correspond to approximations which interpolate the following pairs of data points: (1st, 3rd), (2nd, 3rd), (2nd, 4th), (2nd, 5th), (2nd, 6th), (3rd, 6th), (3rd, 7th), $\cdots$. After 201 iterations, the (unique) best approximation is obtained which interpolates the 51st and 151st data points. The difficulty arises because of the nonnegativity restrictions on u_i, v_i (and also b_j, c_j) which are somewhat artificial. What is required is an algorithm which at each iteration can bypass intermediate data points and thereby substantially reduce the number of iterations required to obtain a best approximation. We shall now describe such an algorithm; it requires no more than 7 iterations to solve the above problem.

Inspection of the linear programming problem (2) reveals that (i) an initial basic feasible solution is immediately available, and (ii) only n columns are needed to store the information contained in the right-hand sides of the equality constraints. Thus, denoting the columns of the simplex tableau corresponding to (2) by $\mathbf{R}$, $\mathbf{b}_j$, $\mathbf{c}_j$, $\mathbf{u}_i$, $\mathbf{v}_i$ (see Table 2, for example), an initial basis is provided by $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_m$ whenever each f_i is nonnegative. If an f_i is negative we change the sign of the corresponding row and replace $\mathbf{u}_i$ in the basis by $\mathbf{v}_i$. It is also clear that $\mathbf{b}_j = -\mathbf{c}_j$ and $\mathbf{u}_i = -\mathbf{v}_i$, and that the sum of the marginal (or reduced) costs of $\mathbf{b}_j$ and $\mathbf{c}_j$ is zero and of $\mathbf{u}_i$ and $\mathbf{v}_i$ is -2. Thus the condensed form of the simplex method (in which the basis is suppressed) can be applied to just n columns, which initially contain $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n$.

The simplex iterations can be performed within an array of dimensions $(m + 2) \times (n + 2)$ which initially contains the data in Table 1 (including labels for the basic and nonbasic vectors). The FORTRAN program [6] also requires additional arrays totaling $3m + n$ words: these are used as workspace and for storing output information.

TABLE 1
Initial condensed simplex tableau for the algorithm, assuming each f_i is nonnegative. (A *full* tableau is displayed in Table 2.)

Basis	**R**	$\mathbf{b}_1$	$\mathbf{b}_2$	$\cdots$	$\mathbf{b}_n$
$\mathbf{u}_1$	f_1	$\varphi_{1,1}$	$\varphi_{2,1}$	$\cdots$	$\varphi_{n,1}$
$\mathbf{u}_2$	f_2	$\varphi_{1,2}$	$\varphi_{2,2}$	$\cdots$	$\varphi_{n,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\mathbf{u}_m$	f_m	$\varphi_{1,m}$	$\varphi_{2,m}$	$\cdots$	$\varphi_{n,m}$
Marginal costs $\rightarrow$	$\sum_{i=1}^{m} f_i$	$\sum_{i=1}^{m} \varphi_{1,i}$	$\sum_{i=1}^{m} \varphi_{2,i}$	$\cdots$	$\sum_{i=1}^{m} \varphi_{n,i}$

The algorithm is implemented in two stages. Stage 1 restricts the choice of pivotal column during the first n iterations to the vectors $\mathbf{b}_j$ and $\mathbf{c}_j$. The vector to enter the basis is chosen as that with the largest nonnegative marginal cost. The vector leaving the basis is chosen from among the basic vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ by selecting that which causes the maximum reduction in the objective function. At the end of Stage 1, the rank k ($\leqq n$) of the matrix Φ is determined by the total number of vectors $\mathbf{b}_j$, $\mathbf{c}_j$ in the basis. Since k of the vectors $\mathbf{u}_i$ (or $\mathbf{v}_i$) have been removed from the basis, the current simplex tableau represents an approximation which interpolates at least k data points. If the approximation interpolates more than k data points, then the simplex tableau is degenerate; this does not cause any problems in practice.

Stage 2 involves interchanging nonbasic $\mathbf{u}_i$ or $\mathbf{v}_i$ with basic $\mathbf{u}_i$ or $\mathbf{v}_i$: the basic $\mathbf{b}_j$ and $\mathbf{c}_j$ vectors are not allowed to leave the basis during Stage 2. The vector entering the basis is that with the most positive marginal cost, and the vector leaving the basis is again chosen as that which causes the maximum reduction in the objective function. The algorithm terminates when all the marginal costs are nonpositive. In Stage 2 each simplex tableau corresponds to an approximation which interpolates k data points (assuming nondegeneracy). At each iteration, $k - 1$ of these points remain fixed. The vector entering the basis determines which point is to be dropped from the interpolating set while the vector leaving the basis determines the new point of interpolation.

The final tableau at the end of Stage 2 may be infeasible since some of the basic vectors $\mathbf{b}_j$ and $\mathbf{c}_j$ can have negative values associated with them. The solution becomes feasible (and hence optimal) by interchanging such basic vectors $\mathbf{b}_j$ (or $\mathbf{c}_j$) with the corresponding nonbasic vectors $\mathbf{c}_j$ (or $\mathbf{b}_j$).

The main modification to the simplex method is in choosing the vector $\mathbf{u}_i$ or $\mathbf{v}_i$ to leave the basis. *In both Stage 1 and Stage 2 this vector is chosen as that which causes the maximum reduction in the objective function.* A direct search over all the basic vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ is inefficient and a computationally more efficient technique can be used. This is illustrated by means of a simple worked example.

Example. Find the best l_1-approximation to $\{(1, 1), (2, 1), (3, 2), (4, 3), (5, 2)\}$ by $L(A, x) = a_1 + a_2 x$. Equivalently, find the l_1-solution to the overdetermined

system of equations:

$$a_1 + a_2 = 1$$
$$a_1 + 2a_2 = 1$$
$$a_1 + 3a_2 = 2$$
$$a_1 + 4a_2 = 3$$
$$a_1 + 5a_2 = 2$$

The full initial simplex tableau is given in Table 2.

TABLE 2
Full initial simplex tableau for the worked example

Costs ↓	→ Basis	R	0 $\mathbf{b}_1$	0 $\mathbf{b}_2$	0 $\mathbf{c}_1$	0 $\mathbf{c}_2$	1 $\mathbf{u}_1$	1 $\mathbf{u}_2$	1 $\mathbf{u}_3$	1 $\mathbf{u}_4$	1 $\mathbf{u}_5$	1 $\mathbf{v}_1$	1 $\mathbf{v}_2$	1 $\mathbf{v}_3$	1 $\mathbf{v}_4$	1 $\mathbf{v}_5$
1	$\mathbf{u}_1$	1	1	1	−1	−1	1					−1				
1	$\mathbf{u}_2$	1	1	2**	−1	−2		1					−1			
1	$\mathbf{u}_3$	2	1	3***	−1	−3			1					−1		
1	$\mathbf{u}_4$	3	1	4	−1	−4				1					−1	
1	$\mathbf{u}_5$	2	1	5*	−1	−5					1					−1
Marginal costs →		9	5	15	−5	−15	0	0	0	0	0	−2	−2	−2	−2	−2

At the first iteration in Stage 1, $\mathbf{b}_2$ is brought into the basis corresponding to the largest marginal cost (=15). The normal simplex pivot (5*) corresponds to an approximation in which $b_2 = 2/5, u_5 = v_5 = 0$, i.e., an approximation which interpolates the fifth data point. However, if we increase b_2 beyond the value 2/5 we can further reduce the objective function, but this makes u_5 negative. Hence we replace $\mathbf{u}_5$ in the basis by $\mathbf{v}_5$. This can be accomplished by subtracting twice the fifth row from the marginal cost row (thus making the marginal cost of $\mathbf{v}_5$ zero), changing the sign of the fifth row and replacing the label $\mathbf{u}_5$ by $\mathbf{v}_5$ in the basis. The marginal cost of $\mathbf{b}_2$ is now 5, and hence we increase b_2 further.

The second pivot (2**) corresponds to an approximation in which $b_2 = 1/2$, $u_2 = v_2 = 0$, i.e., an approximation which interpolates the second data point. Interchanging $\mathbf{u}_2$ and $\mathbf{v}_2$ in the basis reduces the marginal cost of $\mathbf{b}_2$ to 1. Thus we increase b_2 further.

The third pivot (3***) corresponds to an approximation in which $b_2 = 2/3$, $u_3 = v_3 = 0$, i.e., an approximation which interpolates the third data point. The objective function cannot be decreased further by increasing b_2, since if we interchange $\mathbf{u}_3$ and $\mathbf{v}_3$ in the basis, the marginal cost of $\mathbf{b}_2$ becomes −5. Hence we pivot on this element (3***) and bring $\mathbf{b}_2$ into the basis in place of $\mathbf{u}_3$. Using the condensed tableau, the complete solution to this example is given in Table 3.

After two iterations (at the end of Stage 1) the marginal costs of $\mathbf{u}_1$ and $\mathbf{u}_3$ are −1 and 0 respectively, and so the marginal costs of the (suppressed) vectors $\mathbf{v}_1$ and

TABLE 3
Solution to the worked example using condensed tableaux

Basis	R	$\mathbf{b}_1$	$\mathbf{b}_2$
$\mathbf{u}_1$	1	1	1
$\mathbf{u}_2$	1	1	2**
$\mathbf{u}_3$	2	1	3***
$\mathbf{u}_4$	3	1	4
$\mathbf{u}_5$	2	1	5*
Marginal costs	9	5	15

Basis	R	$\mathbf{b}_1$	$\mathbf{u}_3$
$\mathbf{u}_1$	1/3	2/3*	−1/3
$\mathbf{v}_2$	1/3	−1/3	2/3
$\mathbf{b}_2$	2/3	1/3	1/3
$\mathbf{u}_4$	1/3	−1/3	−4/3
$\mathbf{v}_5$	4/3	2/3	5/3
Marginal costs	7/3	2/3	−1/3

Basis	R	$\mathbf{u}_1$	$\mathbf{u}_3$
$\mathbf{b}_1$	1/2	3/2	−1/2
$\mathbf{v}_2$	1/2	1/2	1/2
$\mathbf{b}_2$	1/2	−1/2	1/2
$\mathbf{u}_4$	1/2	1/2	−3/2
$\mathbf{v}_5$	1	−1	2
Marginal costs	2	−1	0

$\mathbf{v}_3$ are -1 and -2 respectively (recall that the sum of the marginal costs of $\mathbf{u}_i$ and $\mathbf{v}_i$ is -2). Since all of the nonbasic vectors have nonpositive marginal costs, there is no need for Stage 2 in this example. The final tableau represents a best approximation $L(A^*, x) = 1/2 + (1/2)x$ which interpolates the first and third data points. The deviations at the other points are given by $v_2 = 1/2, u_4 = 1/2, v_5 = 1$ and the error of approximation is 2.

Further inspection of the final tableau reveals that this best approximation is not unique: notice that $\mathbf{u}_3$ can be brought into the basis yielding a new best approximation. However, in the presence of degeneracy the occurrence of a zero marginal cost in the final tableau is not a sufficient condition for nonuniqueness. (Consequently, the FORTRAN program [6] merely indicates the *possibility* of nonuniqueness in the event that some nonbasic vector in a final tableau has a zero marginal cost.)

Figure 1 contains a block chart of our algorithm: it is simply a modification of the standard form of the simplex method. We conclude this section by summarizing these modifications, assuming throughout that the reader is completely familiar with the standard form of the simplex method (see, for example, Gass [11, Chap. 4]).

(a) Stage 1 refers to the first n iterations and Stage 2 refers to all subsequent iterations.

(b) Initialization of the tableau is accomplished by entering the data corresponding to Table 1 into an array of dimensions $(m + 2) \times (n + 2)$. A negative-vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n$, and the basic vectors $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_m$ are labeled $n + 1$, calculation of the marginal cost row, and $\mathbf{v}_i$ must then replace $\mathbf{u}_i$ in the initial basis. (In the FORTRAN program [6] the labels $1, 2, \cdots, n$ are assigned to the nonbasic vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n$, and the basic vectors $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_m$ are labelled $n + 1$, $n + 2, \cdots, n + m$. The suppressed vectors $\mathbf{c}_j(= -\mathbf{b}_j)$ and $\mathbf{v}_i(= -\mathbf{u}_i)$ are identified, whenever it becomes necessary, by the labels $-1, -2, \cdots, -n$ and $-(n + 1)$, $-(n + 2), \cdots, -(n + m)$, respectively. The user supplies the values f_i and $\varphi_{j,i}$, and the program then deals with any negative-valued f_i's, computes the marginal costs, and records the labels of the vectors which form the initial basis.)

(c) During Stage 1 the vector to enter the basis is chosen from among the vectors $\mathbf{b}_j$ and $\mathbf{c}_j$ as that with the largest nonnegative marginal cost. Recall that the sum of the marginal costs of each pair $\mathbf{b}_j$ and $\mathbf{c}_j$ is zero, and hence the marginal

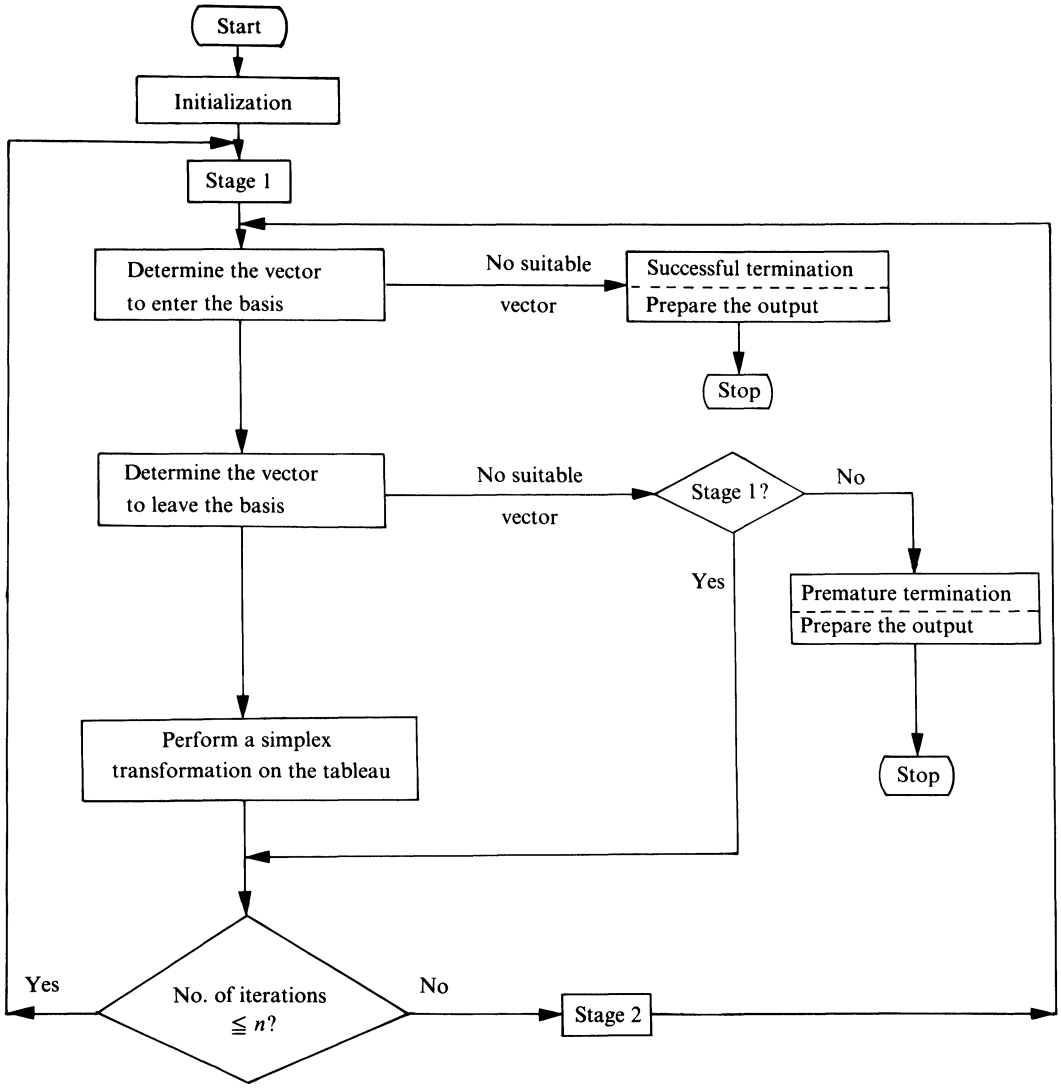

FIG. 1. *A block chart of the algorithm*

costs of all $2n$ of these vectors are always immediately available. During Stage 2 the vector to enter the basis is chosen from among the nonbasic vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ as that with the most positive marginal cost: recall that the sum of the marginal costs of each pair $\mathbf{u}_i$ and $\mathbf{v}_i$ is -2. The algorithm normally terminates during Stage 2 because no suitable vector can be designated as the next pivotal column, i.e., no nonbasic $\mathbf{u}_i$ or $\mathbf{v}_i$ remains with a positive marginal cost. In this event, a solution has been attained. The final tableau may contain basic vectors $\mathbf{b}_j$ or $\mathbf{c}_j$ which have negative values associated with them: in order to yield a feasible solution any such row must be multiplied by (-1), and the basic vector $\mathbf{b}_j$ (or $\mathbf{c}_j$) must be interchanged with the corresponding nonbasic vector $\mathbf{c}_j$ (or $\mathbf{b}_j$). As is shown

in Table 3, the output information is gathered from that column of the final tableau which originally contained the f_i values (the column vector **R** of Table 1).

(d) In both Stage 1 and Stage 2 the vector to leave the basis is chosen from among the basic vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ by selecting that vector which causes the maximum reduction in the objective function. Geometrically, the simplex transformation which then follows is usually equivalent to a movement through several neighboring simplex vertices. The normal rule (see Gass [11, p. 68]) for determining the vector to leave the basis is modified as follows. Firstly, apply the normal rule to determine the pivotal row from among the basic vectors $\mathbf{u}_i$ and $\mathbf{v}_i$. Locate the pivot, i.e., that positive element of the tableau which lies at the intersection of the pivotal column and the pivotal row. If subtracting twice the value of the pivot from the marginal cost of the pivotal column yields a nonpositive result, then proceed to the simplex transformation ((e) below) with this pivot. Otherwise, subtract twice the pivotal row from the marginal cost row, multiply the pivotal row by (-1), and replace in the basis the vector $\mathbf{u}_i$ (or $\mathbf{v}_i$) corresponding to the pivotal row by $\mathbf{v}_i$ (or $\mathbf{u}_i$). This operation decreases the value of the objective function and changes the sign of the pivot. The normal rule for determining the pivotal row is applied again, and a new pivot is located. The procedure is repeated until eventually a pivot is determined which cannot be rejected, and a simplex transformation is then performed with this pivot.

It may happen that no suitable vector can be found to leave the basis, i.e., in the current tableau the chosen pivotal column contains no positive elements opposite the basic $\mathbf{u}_i$'s and $\mathbf{v}_i$'s. In Stage 1 this occurs if the rank of the matrix $\Phi = \{\varphi_{j,i}\}^T$ is less than n. In this event the current pivotal column vector $\mathbf{b}_j$ (or $\mathbf{c}_j$) can be ignored in all future computations, and no simplex transformation is performed in this iteration. In Stage 2, since a solution to the l_1-problem is guaranteed to exist, a suitable pivotal row should always be available for any pivotal column chosen by the rules of (c) above. However, in practice, rounding errors can cause a nonbasic vector to be chosen as a pivotal column even though there are no positive elements in the vector to act as a pivot. Typically, this occurs because the matrix Φ contains elements of widely different magnitudes and serious loss of significance has occurred during some simplex transformations. If this situation arises the algorithm should be terminated prematurely: in such cases we usually run the problem again in multiple precision arithmetic, occasionally after attempting some preconditioning of the matrix Φ.

(In our FORTRAN program [6] we use a small positive tolerance limit below which the magnitude of any quantity is considered to be zero. The algorithm is terminated prematurely if a pivotal column is encountered which contains no candidates for pivot that exceed this tolerance limit. This phenomenon is fairly rare, and even when it does occur it is likely that the current solution is near to the optimal solution. For this reason our program [6] prepares the usual output information based on the current tableau rather than the (true) final tableau.)

(e) The rules for the simplex transformation of the tableau are the normal rules employed in the standard form of the simplex method (see Gass [11, p. 67]).

(f) The iteration counter in the algorithm is increased by one either by performing one simplex transformation or because the simplex transformation is bypassed during Stage 1 because no suitable vector can be found to leave the basis.

4. Final remarks. The algorithm described in this paper permits best l_1-linear approximations to be computed more efficiently than was hitherto possible. An important application arises in data fitting by nonlinear approximating functions, for in this case some linearization of the problem will often be employed which calls for the *repeated* application of a linear algorithm. Two recent methods along this line are described in Barrodale, Roberts and Hunt [7] and Osborne and Watson [12].

Acknowledgments. We are very grateful to Dr. Philip Robers and to Dr. Karl Usow for supplying us with programs for their algorithms. Fred Crary and Bruce Wilson provided us with excellent programming assistance.

REFERENCES

[1] N. N. ABDELMALEK, *Linear L_1 approximation for a discrete point set and L_1 solutions of overdetermined linear equations*, J. Assoc. Comput. Mach., 18 (1971), pp. 41–47.

[2] J. BARRODALE, *L_1 approximation and the analysis of data*, Appl. Statist., 17 (1968), pp. 51–57.

[3] ———, *On computing best L_1 approximations*, Approximation Theory, A. Talbot, ed., Academic Press, New York, 1970, pp. 205–215.

[4] I. BARRODALE AND F. D. K. ROBERTS, *Applications of mathematical programming to l_p approximation*, Nonlinear Programming, J. B. Rosen, O. L. Mangasarian and K. Ritter, eds., Academic Press, New York, 1970, pp. 447–464.

[5] ———, *An improved algorithm for discrete l_1 linear approximation*, TSR 1172, Mathematics Research Center, Madison, Wis., 1972.

[6] ———, *Solution of an over-determined system of equations in the l_1 norm*, Mathematics Dept. Report 69, University of Victoria, Victoria, British Columbia, 1972.

[7] I. BARRODALE, F. D. K. ROBERTS AND C. R. HUNT, *Computing best l_p approximations by functions nonlinear in one parameter*, Comput. J., 13 (1970), pp. 382–386.

[8] I. BARRODALE AND A. YOUNG, *Algorithms for best L_1 and L_∞ linear approximations on a discrete set*, Numer. Math., 8 (1966), pp. 295–306.

[9] G. B. DANTZIG, *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J., 1963.

[10] H. EKBLOM AND S. HENRIKSSON, *L_p-criteria for the estimation of location parameters*, SIAM J. Appl. Math., 17 (1969), pp. 1130–1141.

[11] S. I. GASS, *Linear Programming*, 2nd ed., McGraw-Hill, New York, 1964.

[12] M. R. OSBORNE AND G. A. WATSON, *On an algorithm for discrete nonlinear L_1 approximation*, Comput. J., 14 (1971), pp. 184–188.

[13] P. RABINOWITZ, *Applications of linear programming to numerical analysis*, SIAM Rev., 10 (1968), pp. 121–159.

[14] J. R. RICE AND J. S. WHITE, *Norms for smoothing and estimation*, Ibid., 6 (1964), pp. 243–256.

[15] P. D. ROBERS AND A. BEN-ISRAEL, *An interval programming algorithm for discrete linear L_1 approximation problems*, J. Approximation Theory, 2 (1969), pp. 323–336.

[16] P. D. ROBERS AND S. S. ROBERS, *Discrete linear L_1 approximation by interval linear programming*, Manuscript.

[17] K. H. USOW, *On L_1 approximation II; Computation for discrete functions and discretization effects*, this Journal, 4 (1967), pp. 233–244.

[18] ———, *An algorithm for discrete linear L_1 approximation with application to spline functions*, Manuscript.

[19] H. M. WAGNER, *Linear programming techniques for regression analysis*, J. Amer. Statist. Assoc., 54 (1959), pp. 206–212.