A COMPARISON OF METHODS FOR DISCRETE L_1 CURVE-FITTING

by

D. Anderson and W.L. Steiger

DCS-TR-96

Department of Computer Science
Hill Center for the Mathematical Sciences
Busch Campus
New Brunswick, N.J. 08903

January, 1981

TITLE: A Comparison of Methods for Discrete L_1 Curve-Fitting

AUTHORS: D. Anderson
Department of Mathematics
Faculty of Military Studies
R.M.C. Duntroon
Duntroon, N.S.W. Australia

W. L. Steiger [1]
Department of Computer Science
Rutgers University
New Brunswick, N.J. 08903

KEY WORDS: Least absolute deviation, curve-fitting, linear programming.

RUNNING TITLE: Comparison of Methods

1 Research supported in part by a visiting fellowship in the Department of Statistics, The Australian National University. Computations carried out on the Rutgers Laboratory for Computer Science Research DECSYSTEM 20.

ABSTRACT

We study discrete L_1 curve-fitting of n points in k dimensional space. Execution times for the algorithms of Barrodale-Roberts (BR), Bartels-Conn-Sinclair (BCS), and Bloomfield-Steiger (BS) - three of the best LAD curve-fitting procedures - are compared over a variety of deterministic and random curve-fitting problems. Analysis of the results allows us to make surprisingly precise statements about the computational complexity of these algorithms.

In particular, BR is in a different complexity class than BCS and BS as the number of points, n, increases. All algorithms are linear in the dimension, k, and BS is less complex than BCS.

1 Introduction and Background

In the discrete LAD (or L_1) curve-fitting problem one is given n points $(\underline{x}_i, y_i) \in R^{k+1}$, i=1,...,n, the object being to minimize the distance function

$$(1) \qquad \sum_{i=1}^{n} |y_i - \langle \underline{c}, \underline{x}_i \rangle| \equiv \sum_{i=1}^{n} |y_i - \sum_{j=1}^{k} c_j x_{ij}|.$$

The optimal $\underline{c}$ describes a k-dimensional hyperplane [1] $\pi_c = \{(\underline{x},y): y = \langle \underline{c},\underline{x} \rangle\} \in R^{k+1}$ that fits the n points "best" in the sense of discrete L_1.

Let $x_{ij} = \phi_j(t_i)$, the ϕ_j being k linearly independent basis functions, the t_i, n distinct points in R. In this case the optimal $\underline{c}$ determines $P_k = \sum_{i=1}^{k} c_i \phi_i$, the "best" k-term approximation to f as measured in L_1 over the t_i.

The method seems to have greater antiquity than least squares. Laplace may have developed an algorithm to minimize (1) (E. Seneta, pers. comm.) and certainly Boscovitch (see [13]) knew how to do it when $\sum_{i=1}^{n} y_i = c_1 + c_2 \sum_{i=1}^{n} x_i$; this is the case k=2 with $x_{i1} = 1$, all i, and the fit constrained to pass through the centroid of the data. Gauss knew how to do the k = 2 case (E. Seneta, pers. comm.) and later, Edgeworth [11], [12] dropped the constraint and considered k > 2 as well.

---

1 Throughout we use $\underline{c}$ for the hyperplane, π_c, it determines.

In more recent times, but before the advent of computers, Rhodes [20] and Singleton [23] developed algorithms which, even after the appearance of computers, were not implemented -- they were just too complicated.

The connection with linear programming and the development of Simplex codes led to the first automatic procedures to do LAD curve fits. T. Harris [14] seems to be the first to have mentioned that (1) could be solved by setting up an appropriate LP and using Simplex, but Charnes, Cooper and Ferguson [9] may have been the first to give details. In a similar vein to the derivation in Wagner [27], (1) is equivalent to the primal LP problem

$$\min \sum_{i=1}^{n} r_i$$

$$\text{(2)} \quad \text{subject to} \quad \left.\begin{aligned} r_i + \sum_{j=1}^{k} c_j x_{ij} &\geq y_i \\ r_i - \sum_{j=1}^{k} c_j x_{ij} &\geq -y_i \end{aligned}\right\} \quad i = 1,\ldots,n$$

with $n + k$ variables r_i, $i=1,\ldots,n$, and c_j, $j=1,\ldots,k$, and $2n$ inequality constraints. At optimum, if $X = (x_{ij})$ is of full rank, at least k r_i's will be zero. In addition $r_i = |y_i - \langle \underline{c}, \underline{x}_i \rangle|$, $i = 1,\ldots,n$, and so it turns out that at most $n-k$ of the $2n$ inequalities are unattained.

The LP problem (2) is already much bigger than the L_1 fit (1) that gave rise to it: the data of (1) may be written in an $n \times (k+1)$

table while the data of (2) would require a $2n \times (n+k)$ table. Some abbreviation is possible by passing to the dual. Using the duality theorem, the problem

$$\max\left[\sum_{i=1}^{n} y_i u_i - \sum_{i=1}^{n} y_i v_i\right]$$

$$(3)\quad \text{subject to} \begin{cases} u_i + v_i = 1 & i = 1,\ldots,n \\ \sum_{i=1}^{n} x_{ij} u_i - \sum_{i=1}^{n} x_{ij} v_i = 0 & j = 1,\ldots,k \\ u_i,\ v_i \geq 0 & i = 1,\ldots,n \end{cases}$$

is dual to (2). If the v_i are regarded as slack variables in $u_i \leq 1$, substitution of $v_i = 1 - u_i$ in (3) gives the bounded variable problem

$$\max \sum_{i=1}^{n} y_i u_i$$

$$(4)\quad \text{subject to} \begin{cases} \sum_{i=1}^{n} x_{ij} u_i = \tfrac{1}{2} \sum_{i=1}^{n} x_{ij}, & j = 1,\ldots,k \\ 0 \leq u_i \leq 1, & i = 1,\ldots,n \end{cases}$$

which has only n variables and k equality constraints. Nevertheless, because of the inequalities (4) is larger than the original problem (1) and, as would be expected, direct methods are much faster than solving (4) by Simplex.

In case $k = 1$, (1) becomes

$$(5)\qquad \sum_{i=1}^{n} |y_i - c x_i| \equiv \sum_{i=1}^{n} |x_i| \left| \frac{y_i}{x_i} - c \right|$$

from which the solution is seen to be the weighted median of the ratios y_i/x_i with weights $|x_i|$, or equivalently, the median of the distribution

$$(6) \qquad G(u) = \sum_{A} |x_i| / \sum_{i=1}^{n} |x_i|, \quad A = \{i : \frac{y_i}{x_i} \leq u\} .$$

This means that the optimal $\underline{c}$ is one of the ratios, say y_p/x_p, that the optimizing line (hyperplane) contains one of the points (the pth), and that a residual (the pth term) in (5) is zero.

In the general case, as is well-known [7], [27] at least r of the residuals, $y_i - \langle \underline{c}, \underline{x}_i \rangle$, in (1) vanish, r being the rank of $X = (x_{ij})$, and $\underline{c}$ contains the corresponding r points. This property is exploited by many of the LAD fitting algorithms.

Assuming X to be of full rank, the algorithms move through a sequence $\underline{c}^0, \underline{c}^1, \ldots$ of approximations until the Nth iteration, say, when the procedure terminates with $\underline{c}^N$, an optimal solution of (1). At the i^{th} step the current approximation $\underline{c}^i$ determines a hyperplane containing at least k of the n points; thus $y_j = \langle \underline{c}^i, \underline{x}_j \rangle$, $j = j_1, \ldots, j_k$. One of these points is eliminated and replaced by one of the n-k excluded points, thus determining a new set of k points $(\underline{x}_j, y_j)$, $j = m_1, \ldots, m_k$. The next approximation $\underline{c}^{i+1}$ is chosen to satisfy $y_j = \langle \underline{c}^{i+1}, \underline{x}_j \rangle$, $j = m_1, \ldots, m_k$.

In one of the better algorithms currently available [5], [4], Barrodale and Roberts (BR) use a variant of (2), the primal linear programming formulation of (1). Simplex ideas and a line search control the elimination and replacement phases of each iteration.

Bartels, Conn and Sinclair (BCS) [6] have an efficient LAD algorithm using projection methods to guide elimination. Once a point is selected for elimination, the replacement is done optimally via a weighted median calculation. At the current step suppose $y_j = \langle \underline{c}, \underline{x}_j \rangle$, $j = j_1, \ldots, j_k$, and that $(\underline{x}_q, y_q)$ has already been selected for deletion. Write $B = \{ j : j \neq q, j = j_1, \ldots, j_k \}$ for the indices of the remaining k-1 points and choose $\underline{\mathscr{d}} \in R^k$ such that $\langle \underline{\mathscr{d}}, \underline{x}_j \rangle = 0$, $j \in B$. Then

$$\underline{\theta}_t = \underline{c} + t\,\underline{\mathscr{d}}, \quad t \in R \tag{7}$$

describes a one parameter family of hyperplanes that contain $(\underline{x}_j, y_j)$, $j \in B$, and $\underline{\theta}_o$ is the current fit, $\underline{c}$. For any t,

$$\begin{aligned} f(\underline{\theta}_t) &= \sum_{i=1}^{n} |y_i - \langle \underline{\theta}_t, \underline{x}_i \rangle| \\ &= \sum_{i \notin B} |y_i - \langle \underline{\theta}_t, \underline{x}_i \rangle| \\ &= \sum_{i \notin B} |y_i - \langle \underline{c}, \underline{x}_i \rangle - t \langle \underline{\mathscr{d}}, \underline{x}_i \rangle| \\ &= \sum_{i \notin B} |u_i - t v_i| \end{aligned} \tag{8}$$

where $u_i = y_i - \langle \underline{c}, \underline{x}_i \rangle$ and $v_i = \langle \underline{\mathscr{d}}, \underline{x}_i \rangle$. From (5), the minimum occurs when t is the weighted median of the ratios u_i/v_i, say u_p/v_p, which implies that the optimal replacement of $(\underline{x}_q, y_q)$ is by $(\underline{x}_p, y_p)$.

In [7] Bloomfield and Steiger (BS) also use weighted medians in another part of their algorithm, as a heuristic to select the point that will be eliminated. They compute weighted medians by a partial quicksort method; BCS uses heapsorting for this purpose.

L_1 curve fitting is of interest because of the development of robust and resistant statistical methods (see [2] or [15], e.g.) and it is certainly of practical importance in approximation theory. In addition, it looks to be even more interesting because of the potential for solving LP problems as equivalent L_1 curve fits, as noticed by Steiger [25]. There, a given LP problem with bounded, feasible solutions is transformed into a problem of the form (4) and thus arises from a certain L_1 curve-fit (1). On the average, BS solves the equivalent LAD fit faster than Simplex solves the given LP. For this reason alone it would be useful to better understand the nature of the good L_1 algorithms and to explain any interesting differences and similarities. It is somewhat surprising that no one seems to have made a serious attempt to do so.

Therefore, the present work studies the performance of three L_1 fitting algorithms, Barrodale-Roberts, Bartels, Conn and Sinclair, and Bloomfield-Steiger. They were chosen because they seem to be superior to a number of other procedures [1], [10], [11], [17], [22], [26].

In the next section we describe the basis under which the algorithms were compared. They were tested on some "random" versions of (1), and on some real polynomial approximation problems. The test results are laid out in Section 3 where they are analyzed in some detail. It is surprising that the techniques gave fairly precise statements about the relative computational complexity of the algorithms.

Finally we have included an extensive bibliography relating to discrete L_1 curve fitting. There seems to be a statistical literature stemming from classical work as well as from modern research in robust techniques. At the same time there has been a great deal of work in numerical analysis and other applied mathematics areas. We hope to acquaint both disciplines with these sources.

2 Comparing the Algorithms

The main goal is to understand how BR(n,k), BCS(n,k) and BS(n,k) depend on n and k, each function denoting the computational complexity of that particular algorithm on a particular curve-fitting problem, (1), of fitting n points in R^{k+1}. There does not seem to be an obvious way of addressing this issue directly. Using the Bloomfield-Steiger algorithm for example, one could write

$$BS(n,k) = \sum_{i=1}^{N(n,k)} C_i ,$$

N(n,k) denoting the number of iterations until the algorithm terminates, and C_i the complexity of the i^{th} iteration. As shown in [7]

$$C_i = bnk + S_i$$

b being a constant and S_i the computational cost of sorting to find a weighted median in the i^{th} iteration. While this will vary from step to step, weighted median algorithms are available for which $S_i \leq dn$ so $C_i \leq bnk + dn$ and hence

$$BS(n,k) \leq N(n,k)\ [bnk+dn]$$

and the problem now reduces to that of understanding how the iteration count, N(n,k), depends on n and k.

The situation is similar with BR and BCS, as are the difficulties in dealing with N(n,k). We have avoided the problem of obtaining an analytical expression for N(n,k), and decided instead to carry out our study in an empirical way.

The three algorithms were compared over a variety of curve fitting problems. Some of these were deterministic and some were random (i.e., estimating the coefficients of a regression model from a random sample).

In the deterministic cases we approximate a function f by $P_k \equiv \sum_{j=1}^{k} c_j \phi_j$, a linear combination of k basis functions ϕ_j. The c_j's are chosen to give the discrete L_1 fit:

$$\text{minimize} \quad \sum_{i=1}^{n} \left| f(t_i) - P_k(t_i) \right| , \text{ the } t_i \text{ n distinct}$$

points. In the notation of (1), $y_i = f(t_i)$ and $x_{ij} = \phi_j(t_i)$.

In the first set of examples, the ϕ_j comprise the monomial basis: $\phi_j(t) = t^{j-1}$ and P_k is a polynomial of degree k-1. First $f(x) = e^x$ is approximated at n+1 evenly spaced points in [0,2]; thus $t_j = 2j/n$, $j=0,\ldots,n$. Table 1 gives the CPU times (net of generating y_i and x_{ij}) and iteration counts for each algorithm over a variety of problem sizes. The number of points, n + 1, ranges as n = 50, 100, 150, 200, 400 and 600. k-1, the degree of P_k, ranges as k = 3, 4, 5, 6, 7, 8.

The case k = 2 was avoided because there are special purpose straight line fits of high efficiency [21], [24].

Table 2 contains the results for the next example. Here $f(x) = \sqrt{x}$ on $[0,1]$ and $t_j = j/n$, $j = 0,1,\ldots,n$. The ranges of n and k are as in Table 1.

In these two examples the columns of the matrix $X = (x_{ij}) \equiv (\phi_j(t_i))$ are "nearly" linearly dependent. This allows the possibility that round off errors could become important in the execution of some of the algorithms and thereby seemingly affect their efficiency. Thus, in the third set of examples the ϕ_j are the Legendre polynomials shifted onto $[0,2]$ and, as in Table 1, $f(x) = e^x$. The ranges of n and k are as before. BR and BCS had to work harder to produce the same approximations in the (orthogonal) Legendre basis than in the monomial basis, suggesting that conditioning in the original data matrix can indeed affect performance.

Table 4 depicts the approximation of $f(x)=\sqrt{x}$ on $[0,1]$ by linear combinations of shifted Legendre polynomials. Here BR had a much easier (cheaper) time than in the monomial basis case, while BCS and BS had to work slightly harder to obtain the same approximations.

Finally the three algorithms were compared in L_1 curve fits arising from linear regression models. A sample of size n was generated from the model

$$Y = c_1X_1 + \ldots + c_kX_k + U \tag{9}$$

as follows. For each $i = 1,\ldots,n$, successive random numbers x_{i2}, x_{i3}, $\ldots x_{ik}$, u_i were generated. We then set $x_{i1} = 1$ and formed $y_i = c_1 x_{i1} + \ldots + c_k x_{ik} + u_i$. We chose $c_i = \sqrt{i-1}$ so that the columns of $X = (x_{ij})$ would be scaled differently. Given such a sample of size n from (9) the three algorithms were used to obtain LAD estimates of the c_i. The CPU times and iteration counts were recorded.

The entire process of generation and estimation was performed 25 times for a particular choice of n and k, and for each algorithm the CPU time and iterations (net of generation) were accumulated. In this way, the comparisons did not depend too strongly on a particular small segment of the random number generator.

To investigate possible effects of the distribution of the random variables X_i,U on the efficiency of the algorithms, we performed the Monte-Carlo studies using members of the Pareto family of densities, p_a, where $p_a(t) = a/(1+t)^{1+a}$, $t \geq 0$, $a \geq 0$. If $a>1$ the expectation exists and equals $a/(a-1)$ and thus the density

$$(10) \qquad p_a^*(t) = a/[1+(t-c)]^{1+a}, \quad a \geq 0, \quad t \geq c = -a/(a-1)$$

is Pareto and is centered at its mean (of zero).

If $a>2$ p_a has finite variance and as a rule, the bigger a, the tighter around $a/(a-1)$ are the values of a random variable following p_a. We present results of Monte-Carlo studies for $a = 1.2$ and $a = 2.2$. In the first case p_a has infinite variance and the random numbers are

very "spread out". In the latter, the variance is finite and the random numbers are more tightly clustered about their mean.

Table 5 contains the results of the experiments for a = 1.2. It presents the total CPU time and total iteration counts (over the 25 repetitions) for each algorithm. In it n ranges through 100, 200, 300, 600, 900, while k takes the values 3, 4, 5, 6, 7, 8.

Table 6 contains the results of L_1 curve fits arising from the regression in (9) where the variables are centered Pareto with a = 2.2. Finally, in Table 7, regressions with unit normal variables are depicted. The ranges of n and k are as in Table 5.

The computations were performed on a DECSYSTEM-2060. The FORTRAN implementation of Barrodale-Roberts [5] was used for BR and a FORTRAN version supplied by the authors was used for BCS. An unpublished FORTRAN code was used for BS.

The random numbers were generated using the FORTRAN function RAN on the DEC 20. In Tables 5, 6, and 7 each cell (n,k) uses a different segment of the generator. But within a cell, each algorithm solved the fit generated by the same random numbers.

Random numbers with density p_a^* have distribution function

$$P_a^*(t) = 1 - 1/[1+(t-c)]^a,$$

Hence, if U is uniform [0,1]

$$U^{-1/a} + C - 1$$

will have centered, Pareto density, (10). This was the method used in

Tables 5 and 6. Normal random numbers were approximated by sums of 12 independent uniforms, less the constant 6.

3. Results and Conclusions

In this section we discuss some of the information contained in Tables 1-7. Although this material may be complicated, it is also very interesting. The analysis will allow us to make reasonably precise statements about the behavior of BR(n,k), BCS(n,k) and BS(n,k), the computational complexity of the algorithms as portrayed by CPU time.

Examination of the columns of all tables reveals a characteristic difference between BR and the other two algorithms. For all k, as n increases, BR increases faster than linearly (like nlogn or n^2), while BCS and BS increase linearly. The following graph (Figure 1) of complexity versus n is based on the data in Table 4 with k=3. The straight lines drawn through the points (n, BCS(n,3)) and (n,BS(n,3)) were obtained as linear LAD fits to these data and are extremely close to the actual CPU times.

[PUT FIGURE 1 HERE]

The curves in the graph are typical of those for all k, in all tables. This fact expresses an important difference in the complexity as a function of number of points being fit and is our first conclusion:

$$\text{(11)} \qquad \begin{aligned} BCS(n,k)/BR\ (n,k) &\longrightarrow 0 \\ BS(n,k)/BR\ (n,k) &\longrightarrow 0 \end{aligned} \qquad \text{as } n \longrightarrow \infty$$

It shows that BCS is asymptotically infinitely superior to BR, contrary to the belief of the authors in [6].

A more detailed analysis for BCS and BS is suggested because the CPU times are so linear in n. LAD straight line fits of the data in each of the columns of Tables 1-7 were performed. The data in Tables 5-7 were first divided by 25 to make the timings comparable to those in Tables 1-4. In all cases, CPU times were extremely well described in the sense that departures from the linear fits were neglegible. The observation leads to our second assertion

$$
\begin{aligned}
BCS(n,k) &= a_k(f)n + b_k(f) \\
BS(n,k) &= a'_k(f)n + b'_k(f) \quad ,
\end{aligned}
\tag{12}
$$

that the complexity of these algorithms is linear in n. The slopes $a_k(f)$ and $a'_k(f)$ depend on the number of parameters, k, being fit and on f, a variable that describes the curve-fitting problem (eg. e^x approximated by k^{th} degree polynomials, k^{th} order regression with Pareto (a=1.2) errors, etc.)

Table 8 shows $a_k(f)$ and $a'_k(f)$ for all the values of k that were considered and in each curve-fitting problem that was studied. In all cases, $a'_k(f) < a_k(f)$, and the difference is usually sizeable. For example, when k=3 and one is approximating e^x in the Legendre basis (Table 3), BS grows with n at less than 1/3 the rate that BCS does. In general then, we are led to our third assertion:

$$(13) \qquad \frac{BS(n,k)}{BCS(n,k)} \longrightarrow c_k(f) < 1 \quad \text{as } n \longrightarrow \infty \ ,$$

c_k (f) a constant depending on k and the curve-fitting problem (f).

BR(n,k), BCS(n,k) and BS(n,k) all increase with k in a linear fashion, as can be seen by graphing the rows in each table against k. While LAD straight lines do not fit these data quite as well as they did with the plots of CPU time versus n, the fits are still excellent. However, in the deterministic context (Tables 1-4) there is a slight tendency for all the algorithms to work proportionately harder for even k then for odd k. There seems to be no ready explanation for this curious fact.

Dividing the data in Tables 5-7 by 25 so they are comparable to the CPU times for the deterministic context portrayed in Tables 1-4, one sees that all the algorithms had to work harder to obtain the regression fits than the deterministic ones. The Pareto (a = 1.2) case was the easiest regression and BR and BS had the most difficulty with the Normal case (Table 7). This may be due to the fact that $(\underline{x}_i, y_i)$ points are highly dispersed in the former case. In general, the k points determining the optimal hyperplane will tend to be "spread out", with high inter point distances. With Normal data many sets of k points will have nearly equal, high dispersion, and define near-optimum fits. But with infinite variance Pareto data, only few sets of k points are nearly optimal so the task of discriminating is simpler in this case.

In approximating e^x (Tables 1, 3) BR had to work much harder in the Legendre basis, and BCS a little harder; BS did not prefer one basis to the other. On the other hand, in approximating $\sqrt{x}$, BR had to work much harder in the monomial basis, while the other algorithms had a slightly more difficult time in the Legendre basis. There may be an explanation for these observations based on the distribution of the data in these deterministic fitting problems and on the exchange criteria used by each algorithm, but none has offered itself thus far.

Finally, based on (12) and the linearity of all the algorithms in k, we propose the following description of BCS and BS:

$$\begin{aligned} BCS(n,k) &= a + bn + ck + dnk \\ BS(n,k) &= a' + b'n + c'k + d'nk \end{aligned} \tag{14}$$

For each table, values for the parameters in (14) were obtained via LAD fits which, in the case of BS for example , minimize,

$$\sum_{n,k} \left| BCS(n,k) - (a + bn + ck + dnk) \right| \quad .$$

In this way, the complexity of the algorithms in fitting a regression model with Pareto (a = 1.2) errors as shown by the data in Table 5 may be described by

$$\begin{aligned} BCS(n,k) &= .0546 - .0019n - .0170k + .0012nk \\ BS(n,k) &= .0442 - .0017n - .0129k + .0007nk \end{aligned} \tag{15}$$

Both models fit the table data extremely well. We use an L_1 analogue of the R^2 measure from least squares regression to assess the

quality of the fit: In Table 5, the median CPU time for BCS is 1.273 and

$$\sum_{n,k} \left| BCS(n,k) - 1.273 \right| = 34.768$$

measures the variability of the CPU times for the BCS algorithm on the Pareto (a=1.2) regression problems: it is, in fact, an L_1 analogue of standard deviation. The variability measure of the difference between BCS and the right hand side of (15) is 2.36 so the model in (15) may be said to explain 93.21% of the variability of the CPU times. In a similar way the model for BS in (15) explains 91.38% of the variability of the data in Table 5.

The parameter values for the models in (14) are given in Table 9, along with the percent of CPU time variability that is explained by that model. This is done for each curve-fitting problem summarized in Table 1 - Table 7. The information accounts for the linearity of the algorithms in k and in n (see (12), (13)). Furthermore it shows what the relative complexity of the algorithms are as n and k get large simultaneously. In this case, if k is a constant, c, times n,

$$(16) \qquad \begin{aligned} BCS(n,k)/(n,k) &\longrightarrow d(f) \\ BS(n,k)/(n,k) &\longrightarrow d'(f) \end{aligned}$$

f denoting the curve-fitting problem, and d the parameter in Table 9. Finally then,

$$(17) \qquad BCS(n,k)/BS(n,k) \longrightarrow h(f) > 1$$

as $nk \to \infty$ with $k = cn$. For the data in Table 1, for example, $h(f) = 125/89 \simeq 1.404$, showing that BS is asymptotically 40% more efficient than BCS in approximating e^x in the monomial basis. Similar comparisons may be made for all the curve-fitting problems studied.

Finally a model for BR like that in (14) was sought. A model of the form $a + bn + ck + dnk + en^2$ seemed to fit the data in Tables 1-7 most closely. The fit is even better than those shown in Table 9 for BS and BCS, where inclusion of the term en^2 into (14) did not improve the models' descriptive power. We therefore suggest that

$$(18) \qquad BR(n,k)/(kn^2) \longrightarrow h(f)$$

a constant that depends on the curve-fitting problem, f, as $nk \to \infty$, with $k = cn$.

References

[1] Abdelmalek, N.N. (1971). Linear L_1 approximation for a discrete point set and L_1 solutions of overdetermined equations. JACM 18, 41-47

[2] Andrews, David (1974). A robust method for multiple linear regression. Technometrics 16, 523-532

[3] Appa, G. and Smith, C. (1973). On L_1 and Chebyshev estimation. J. Math. Prog. 5, 78-87

[4] Barrodale, I. and Roberts, F.D.K. (1973). An improved algorithm for discrete L_1 linear approximation. SIAM. J. Numer. Anal. 10, 839-848

[5] (1974). Algorithm 478: Solution of an overdetermined system of equations in the L_1 norm. Comm. A.C.M. 14, 319-320

[6] Bartels, R., Conn, A. and Sinclair, James W. (1978). Minimization techniques for piecewise differentiable functions: the L_1 solution to an overdetermined system. SIAM J. Numer. Anal. 15, 224-241

[7] Bloomfield, P. and Steiger, W.L. (1980). Least absolute deviations curve-fitting. SIAM J. Scientific and Statistical Comp. 1, 290-301.

[8] (1980). Remark on a paper of Narula and Wellington. Technometrics 22, 450.

[9] Charnes, A., Cooper, W.W. and Ferguson, R.O. (1955). Optimal estimation of executive compensation by linear programming. Mgt. Sci. 1 138-151

[10] Claerbaut, Jon and Muir, Francis (1973). Robust modelling with erratic data. Geophysics 38, 826-844

[11] Edgeworth, F.Y. (1887). A new method of reducing observations relating to several quantities. Phil. Mag. (Fifth Series) 24, 222-223

[12] (1888). On a new method of reducing observations relating to several quantities. Phil. Mag. (Fifth Series) 25, 184-191

[13] Eisenhart, C. (1961). Boscovitch and the combination of observations. In Roger Joseph Boscovitch, L.L. Whyte, Ed. New York: Fordham University Press

[14] Harris, T. (1950). Regression using minimum absolute deviations. Amer. Statistician 4, 14-15

[15] Huber, Peter J. (1973). Robust Regression, asymptotics, conjectures and Monte-Carlo. Ann. Stat. 1, 799-821

[16] Karst, Otto J. (1958). Linear curve fitting using least deviations. J. Amer. Stat. Assoc 53, 118-132

[17] McCormick, G.F. and Sposito, V.A. (1976). Using the L_2 estimation in L_1 estimation. SIAM J. Numer. Anal. 13, 337-343

[18] Narula, S.C. and Wellington, J.F. (1973). Algorithm AS 108. Multiple linear regression with minimum sum of absolute errors. Applied Stat. 26, 106-111

[19] (1977). Prediction, linear regression and the minimum sum of relative errors. Technometrics 19, 185-190

[20] Rhodes, E.C. (1930). Reducing observations by the method of minimum deviations. Phil. Mag. (Seventh Series) 9, 974-992

[21] Sadovski, A.N. (1974). Algorithm AS74. L_1 - norm fit of a straight line. Applied Stat. 23, 244-248

[22] Schlossmacher, E.J. (1973). An iterative technique for absolute deviations curve fitting. J. Amer. Stat Assoc. 68, 857-865

[23] Singleton, R.R. (1940). A method for minimizing the sum of absolute values of deviations. Ann. Math. Stat. 11, 301-310

[24] Sposito, V. (1976). Remarks on Algorithm AS74. Applied Stat. 25, 96-97

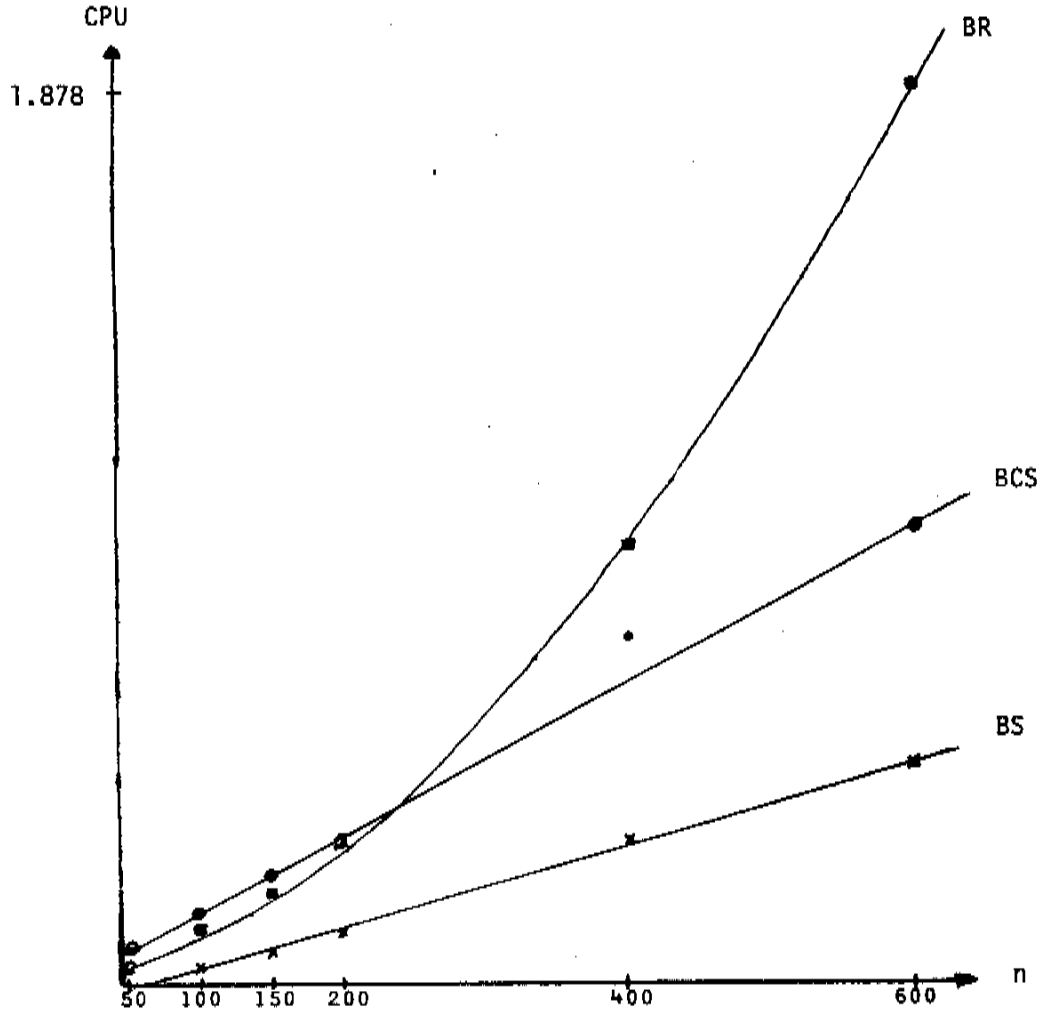
[25] Steiger, W.L. (1980). Linear programming via L_1 curve fitting beats Simplex. Abstracts, AMS, 80T-C26, 385-386

[26] Usow, K.H. (1967). On L_1 approximation II: Computation for discrete functions and discretization effects. SIAM J. Numer. Anal. 4, 233-244

[27] Wagner, Harvey, M. (1959). Linear programming techniques for regression analysis. J. Amer. Stat. Assoc. 54, 206-212

FIGURE 1

BR(■), BCS(o), and BS(x) plotted against n for the data in Table 4, k=3



BCS(n,k) .00156n+.01591
BS(n,k) .00077n-.00640

TABLE 1

Comparison of CPU times and iteration counts for L_1 approximations of e^x on [0,2] by $\sum_{j=1}^{k} c_j t^{j-1}$ based on n + 1 points $t_i = 2i/n$, $i = 0,1,\ldots,n$.

		3		4		5		6		7		8	
		CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
	BR	.043	5	.062	7	.100	10	.149	14	.181	17	.211	16
50	BCS	.121	7	.238	13	.232	11	.356	15	.407	16	.458	17
	BS	.037	6	.080	11	.107	12	.136	13	.157	13	.229	17
	BR	.109	6	.149	8	.221	9	.331	13	.406	17	.494	17
100	BCS	.223	7	.427	13	.421	11	.570	13	.606	13	.787	16
	BS	.071	6	.163	12	.188	11	.296	15	.339	15	.458	18
	BR	.188	6	.236	7	.398	10	.616	15	.650	16	.905	20
150	BCS	.328	7	.620	13	.696	13	.818	13	.968	15	1.513	22
	BS	.103	6	.239	12	.299	12	.432	15	.502	15	.779	21
	BR	.281	6	.351	7	.598	10	.885	14	.986	17	1.339	20
200	BCS	.432	7	.813	13	.958	14	1.183	15	1.141	13	2.134	26
	BS	.138	6	.317	12	.392	12	.703	19	.699	16	1.021	21
	BR	.845	7	1.083	9	1.832	12	2.688	16	2.656	17	3.755	22
400	BCS	.850	7	1.586	13	1.854	14	2.431	16	2.456	15	3.870	24
	BS	.307	7	.673	13	.888	14	1.476	20	1.484	17	2.139	22
	BR	1.643	7	2.063	9	3.509	11	5.242	16	5.267	20	7.311	23
600	BCS	1.271	7	1.857	10	2.772	14	3.279	14	3.822	16	5.923	25
	BS	.458	7	1.069	14	1.254	13	2.198	20	2.207	17	3.063	21

TABLE 2

Comparison of CPU time and iteration counts for L_1 approximation of $\sqrt{x}$ on [0,1] by $\sum_{j=1}^{k} c_j t^{j-1}$ based on n + 1 points $t_i = i/n$, $i = 0,1,\ldots,n$.

		3		4		5		6		7		8	
		CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
50	BR	.067	6	.102	10	.140	13	.167	14	.176	13	.228	16
	BCS	.098	5	.197	10	.237	11	.284	12	.357	14	.417	15
	BS	.038	6	.075	10	.077	8	.129	12	.169	14	.205	15
100	BR	.142	5	.238	10	.280	10	.425	16	.480	15	.541	16
	BCS	.205	6	.412	12	.463	12	.509	12	.736	17	.811	17
	BS	.072	6	.131	9	.175	10	.246	12	.322	14	.434	17
150	BR	.300	6	.445	10	.580	13	.767	17	.836	17	1.001	19
	BCS	.299	6	.557	11	.667	12	.829	14	1.107	18	1.315	20
	BS	.107	6	.193	9	.259	10	.356	13	.447	13	.703	19
200	BR	.413	5	.651	10	.744	9	1.175	18	1.285	15	1.488	18
	BCS	.392	6	.777	12	.880	12	1.195	16	1.504	19	1.578	18
	BS	.141	6	.255	9	.342	10	.577	15	.590	13	.845	17
400	BR	1.607	7	2.158	10	2.738	13	3.387	17	2.633	16	4.435	21
	BCS	.771	6	1.515	12	1.916	14	2.212	15	2.693	17	3.295	20
	BS	.278	6	.546	10	.729	11	1.138	15	1.240	14	1.834	19
600	BR	2.806	5	4.316	12	4.823	10	6.825	18	7.877	17	9.023	20
	BCS	1.154	6	2.258	12	2.868	14	3.287	15	4.339	19	4.888	20
	BS	.420	6	.815	10	1.088	11	1.699	15	2.200	17	2.867	20

TABLE 3

Comparison of CPU times and iteration counts for L_1 approximation of e^x on [0,2] by $\sum_{j=1}^{k} c_j L_j(t)$, based on n + 1 points $t_i = 2i/n$, i=0,1,...,n, L_j the j^{th} shifted Legendre polynomial

		3		4		5		6		7		8	
		CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
50	BR	.063	6	.095	10	.124	11	.180	15	.177	13	.308	25
	BCS	.118	6	.205	10	.225	10	.295	12	.289	11	.498	18
	BS	.037	6	.079	11	.107	12	.136	13	.157	13	.229	17
100	BR	.158	6	.222	10	.291	11	.411	14	.426	14	.537	17
	BCS	.229	6	.377	10	.416	10	.572	13	.661	14	.769	15
	BS	.071	6	.163	12	.188	11	.295	15	.339	15	.501	20
150	BR	.286	6	.389	10	.550	13	.717	14	.764	16	1.031	20
	BCS	.339	6	.587	11	.595	10	1.019	17	1.095	16	1.431	20
	BS	.102	6	.239	12	.299	12	.431	15	.501	15	.652	17
200	BR	.408	6	.653	10	.764	12	1.028	16	1.150	17	1.405	20
	BCS	.437	6	.830	12	.832	11	1.443	19	1.571	18	1.950	21
	BS	.138	6	.317	12	.391	12	.703	19	.697	16	.970	20
400	BR	1.329	7	2.104	12	2.249	12	3.209	15	3.903	17	4.554	25
	BCS	.951	7	1.410	10	1.633	11	2.802	19	2.895	17	3.552	19
	BS	.306	7	.672	13	.887	14	1.475	20	1.483	17	2.063	21
600	BR	2.725	7	4.263	12	4.457	12	5.953	17	6.419	19	8.531	25
	BCS	1.433	7	2.430	12	2.441	11	4.174	19	4.720	19	7.36	30
	BS	.475	7	1.067	14	1.252	13	2.195	20	2.198	17	3.192	22

TABLE 4

Comparison of CPU times and iteration counts for L_1 approximation of x on $[0,1]$ by $\sum_{j=1}^{k} c_j L_j(t)$, based on $n+1$ points $t_i = 2i/n$, $i=0,1,\ldots,n$, L_j the j^{th} shifted Legendre polynomial

		3		4		5		6		7		8	
		CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
50	BR	.044	4	.085	9	.090	8	.147	14	.178	14	.221	18
	BCS	.094	5	.159	8	.259	12	.238	10	.381	15	.514	19
	BS	.037	6	.080	11	.107	12	.136	13	.157	13	.228	17
100	BR	.114	5	.198	9	.188	6	.360	16	.393	14	.488	18
	BCS	.172	5	.289	8	.493	13	.489	12	.714	17	.860	18
	BS	.070	6	.163	12	.187	11	.295	15	.339	15	.474	19
150	BR	.189	4	.345	9	.385	9	.525	14	.692	15	.782	17
	BCS	.250	5	.492	10	.752	14	.704	12	.965	15	1.294	20
	BS	.103	6	.240	12	.300	12	.431	15	.496	15	.795	22
200	BR	.308	5	.524	9	.578	9	.911	17	.992	14	1.211	19
	BCS	.329	5	.595	9	.934	13	.912	12	1.391	18	1.808	21
	BS	.137	6	.319	12	.391	12	.702	19	.697	16	.970	20
400	BR	.932	5	1.600	10	1.690	9	2.500	17	3.007	16	3.435	19
	BCS	.723	6	1.250	10	1.919	14	1.996	14	2.572	16	3.219	20
	BS	.309	7	.678	13	.891	14	1.461	20	1.462	17	2.360	25
600	BR	1.878	5	3.230	11	3.354	9	4.757	17	5.996	16	6.665	22
	BCS	.953	5	1.730	9	2.851	14	2.965	14	3.962	17	4.593	18
	BS	.452	7	1.067	14	1.243	13	2.167	20	2.269	18	3.792	27

TABLE 5

Comparison of CPU times and iteration counts summed over 25 independent repetitions of fitting the model (9), X's and U Pareto (a = 1.2), distributed.

		3		4		5		6		7		8	
		CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
100	BR	2.329	133	3.525	199	4.845	258	6.265	307	7.736	362	9.579	421
	BCS	4.802	150	6.315	185	8.473	235	10.634	276	15.618	385	18.165	427
	BS	1.634	130	2.835	189	4.280	243	5.807	287	8.164	359	9.929	388
200	BR	5.810	148	8.712	230	11.406	277	14.757	341	17.677	387	23.150	477
	BCS	8.800	142	13.849	218	17.267	257	23.305	329	29.228	392	35.409	454
	BS	3.448	143	5.993	208	9.250	274	11.957	304	15.599	351	20.912	422
300	BR	10.716	165	15.685	240	21.072	303	25.628	369	30.722	410	35.965	465
	BCS	14.919	169	19.955	215	28.264	289	33.765	325	41.919	383	55.079	487
	BS	5.904	170	9.416	221	13.581	270	18.593	321	24.724	376	31.094	422
600	BR	30.763	181	39.922	226	51.898	296	68.659	392	81.507	454	96.727	537
	BCS	29.864	172	40.061	219	56.184	298	67.741	335	88.994	422	109.205	500
	BS	11.692	170	18.580	220	28.421	287	37.661	328	52.287	404	67.960	470
900	BR	54.228	173	73.797	232	91.825	297	114.026	366	141.273	474	163.161	509
	BCS	42.613	161	60.526	224	81.709	289	104.158	347	137.131	434	164.120	508
	BS	16.898	162	27.601	220	42.545	287	56.133	327	76.461	395	98.191	452

TABLE 6

Comparison of CPU times and iteration counts summed over 25 independent repetitions of fitting the model (9), X's and U Pareto (a = 2.2) distributed.

		3		4		5		6		7		8	
		CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
100	BR	2.516	166	3.272	191	4.781	270	7.132	392	8.445	418	10.952	510
	BCS	4.861	153	7.154	215	9.872	282	13.159	353	16.055	401	18.235	431
	BS	1.869	156	2.876	194	4.827	282	7.038	357	8.907	396	10.933	432
200	BR	5.495	155	8.727	255	11.678	307	16.454	419	19.283	453	24.793	584
	BCS	9.776	162	15.292	249	18.882	289	29.804	440	33.701	466	40.671	534
	BS	3.729	160	6.744	242	9.128	272	14.626	383	18.139	415	25.448	523
300	BR	10.061	179	14.932	254	19.572	317	25.294	394	31.669	458	43.595	620
	BCS	15.861	184	23.085	256	34.307	370	40.484	404	49.944	471	61.188	556
	BS	5.924	173	10.056	242	15.088	306	22.545	398	29.633	460	39.100	541
600	BR	27.896	191	41.928	302	53.599	366	70.324	463	86.768	550	109.447	691
	BCS	31.951	188	52.694	309	69.280	384	85.736	442	107.685	530	129.927	615
	BS	12.249	182	23.181	287	32.547	336	50.467	454	61.748	486	86.095	606
900	BR	53.620	205	77.816	294	100.714	392	124.573	479	159.659	618	207.675	810
	BCS	53.679	220	76.746	298	94.919	349	140.739	498	174.045	582	199.885	640
	BS	19.799	201	31.478	258	50.909	354	69.596	417	98.602	522	145.888	692

TABLE 7

Comparison of CPU times and iteration counts summed over as independent repetitions of fitting the model (9), X's and U distributed as unit normal.

		3		4		5		6		7		8	
		CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
100	BR	2.628	189	4.532	309	6.051	371	7.675	430	10.414	545	13.102	639
	BCS	4.763	156	6.586	203	10.390	309	13.748	379	15.691	401	20.017	487
	BS	2.076	179	3.511	248	5.669	339	8.265	427	11.301	513	15.765	640
200	BR	6.501	220	10.645	341	14.967	431	21.429	585	27.450	700	31.739	747
	BCS	8.873	153	14.128	238	19.490	312	27.199	410	33.629	475	42.192	569
	BS	4.057	180	8.116	300	12.426	384	20.096	541	26.726	630	33.943	711
300	BR	11.397	244	18.377	356	24.702	466	34.960	598	44.996	720	56.087	852
	BCS	13.729	163	22.064	257	30.940	344	41.183	427	52.011	510	61.594	575
	BS	6.365	192	12.276	305	21.034	443	30.295	549	41.563	661	55.173	780
600	BR	29.117	246	51.092	423	71.403	568	94.026	708	115.517	824	148.900	1020
	BCS	28.032	172	44.059	264	65.065	377	89.161	485	110.459	564	137.302	674
	BS	14.256	223	29.182	372	44.597	478	67.771	625	91.962	742	125.849	904
900	BR	56.600	273	92.479	442	130.753	614	164.854	727	202.856	868	258.478	1072
	BCS	46.146	195	65.847	265	100.534	394	129.853	475	153.596	527	204.203	679
	BS	22.570	238	43.286	370	71.677	516	105.188	651	142.546	772	195.548	941

TABLE 8

Slopes of the straight-line relationships of BCS(n,k) and BS(n,k) with n for various k. (See eq. 8)

		3	4	5	6	7	8
Table 1	BCS	.00210	.00382	.00461	.00531	.00643	.00994
	BS	.00077	.00170	.00212	.00375	.00374	.00511
Table 2	BCS	.00191	.00372	.00481	.00546	.00721	.00813
	BS	.00070	.00137	.00184	.00285	.00306	.00487
Table 3	BCS	.00241	.00410	.00404	.00701	.00806	.01319
	BS	.00078	.00170	.00212	.00374	.00372	.00556
Table 4	BCS	.00156	.00284	.00471	.00496	.00651	.00733
	BS	.00076	.00172	.00210	.00369	.00393	.00642
Table 5	BCS	.00189	.00270	.00366	.00468	.00617	.00730
	BS	.00076	.00123	.00191	.00252	.00345	.00442
Table 6	BCS	.00251	.00351	.00425	.00638	.00802	.00910
	BS	.00092	.00143	.00239	.00314	.00460	.00688
Table 7	BCS	.00213	.00296	.00463	.00587	.00686	.00926
	BS	.00106	.00201	.00338	.00486	.00662	.00923

TABLE 9

Parameters fit to the complexity model (14) for each curve fitting context.

Table (f)		a	b	c	d	Percent Explained
1	BCS	.0100	-.0017	.0016	.00125	84.2
	BS	.0350	-.0019	-.0123	.00089	87.8
2	BCS	.0844	-.0019	-.0281	.00142	80.5
	BS	.0415	-.0019	-.0171	.00092	87.4
3	BCS	.0196	-.0019	-.0015	.00126	87.7
	BS	.0395	-.0017	-.0104	.00077	89.4
4	BCS	-.0398	-.0020	.0163	.00121	90.4
	BS	.0471	-.0022	-.0178	.00098	84.2
5	BCS	.0546	-.0019	-.0170	.00115	93.2
	BS	.0442	-.0017	-.0129	.00074	91.4
6	BCS	.0362	-.0019	-.0263	.00135	94.3
	BS	.0928	-.0027	-.0264	.00102	87.1
7	BCS	.1297	-.0031	-.0372	.00153	93.7
	BS	.1635	-.0040	-.0555	.00151	89.9