L_1-*Statistical Procedures and Related Topics*
IMS Lecture Notes – Monograph Series (1997) Volume 31

Exact algorithms for computing the least median of squares estimate in multiple linear regression

José Agulló

Universidad de Alicante, Spain

Abstract: We propose two finite algorithms to compute the exact least median of squares (LMS) estimates of parameters of a linear regression model with p coefficients. The first algorithm is similar to Stromberg's (1993) exact algorithm. It is based on the exact fit to subsets of p cases and uses impossibility conditions to avoid unnecessary calculations. The second one is based on a branch and bound (BAB) technique. Empirical results suggest that the proposed algorithms are faster than the finite exact algorithms described earlier in the literature.

Key words: Branch and bound, exact algorithms, high breakdown regression, least median of squares, robust regression

AMS subject classification: 62F35, 62J05, 65U05.

1 Introduction

In this paper we consider the multiple linear regression model

$$Y = Z\theta + \epsilon, \tag{1}$$

where Y is an $n \times 1$ vector of dependent variables, θ is a $p \times 1$ vector of unknown parameters, Z is an $n \times p$ design matrix of predictors, and ϵ is an $n \times 1$ vector of true residuals. We denote the ith component of Y and the ith row of Z by y_i and z_i^t, respectively. We suppose that the design matrix Z is fixed and has rank p. Sometimes we also assume that any $p \times p$ submatrix of Z is nonsingular; in this case we say that Z verifies the *Haar condition*, or that the observations are in *general position*. An estimate of θ, say $\hat{\theta}$, gives n residuals $e_i(\hat{\theta}) = y_i - z_i^t\hat{\theta}$. The most well-known

estimator of θ is the Least Squares (LS) estimator. The LS estimator is optimal in several situations but it is severely affected by outliers. It also suffers from the *problem of masking*, which occurs when a data set contains multiple outliers and, at the same time, these outliers are not detected by the usual LS diagnostic procedures. To get a reliable outlier detection and estimation, a high breakdown point estimator should be used. Such an estimator is the *Least Median of Squares* (LMS) estimator, introduced by Rousseeuw (1984). The LMS estimator, which we denote by $\hat{\theta}_{LMS}$, is defined by

$$\hat{\theta}_{LMS} = \text{argmin}_\theta \{|e_i(\theta)|\}_{h:n},$$

where the subscript $h{:}\,n$ denotes the hth order statistic of n numbers. To achieve the maximum breakdown point when the observations are in general position, the coverage h is chosen as $h = [n/2]+[(p+1)/2]$, where $[\,]$ denotes the greatest integer function. For $h = n$, the LMS estimator is equal to the *Chebyshev estimator*, denoted by $\hat{\theta}_C$, and defined by

$$\hat{\theta}_C = \text{argmin}_\theta \{\max_i |e_i(\theta)|\}.$$

The minimal value of the objective function which defines $\hat{\theta}_C$ will be called the *Chebyshev criterion.*

The computation of the LMS estimate is difficult. The most widely used algorithm to approximate the LMS estimate is the PROGRESS algorithm (Rousseeuw and Leroy, 1987). This algorithm computes the exact fit of several *elemental sets* (subsets of size p) of the data set. The exact fit with smallest hth absolute residual gives an approximate LMS estimate. The considered elemental sets can be either all $\binom{n}{p}$ possible such sets, or a random subsample of them. If the regression model has intercept, the intercept of each exact fit can be adjusted to yield a smaller hth absolute residual. For a simple regression model ($p = 2$), Steele and Steiger (1986) prove that an algorithm which examines all elemental sets and adjusts the intercept for each of them, obtains the exact LMS estimate. However, for p greater than 2 such an algorithm does not necessarily yield the exact LMS estimate.

Stromberg (1993) proposes an exact algorithm based on the fact that the exact LMS estimate is a Chebyshev estimate for some subset of size $p + 1$ of the data. Any set of $p + 1$ indices from $\{1, \ldots, n\}$ will be called here a *reference set.* Stromberg's algorithm examines all reference sets, computes the Chebyshev estimate for each of them, and then sets $\hat{\theta}_{LMS}$ as the Chebyshev estimate with smallest hth absolute residual.

In this paper we develop two algorithms to compute the exact LMS estimate that are computationally feasible for small or moderate size samples.

Section 2 describes some properties of the Chebyshev fit and discusses its computation. In Section 3 we propose an algorithm to find the optimal reference set. This algorithm differs from Stromberg's in some important features: It is based on exact fits of elemental sets; it uses accelerations based on impossibility conditions, and it also avoids completely sorting of residuals. In Section 4 we propose a finite exact algorithm based on a *branch and bound* (BAB) technique. The BAB algorithm finds the subset of size h whose Chebyshev estimate gives the exact LMS estimate without exhaustive enumeration of all h-subsets. We describe the basic BAB algorithm and some strategies to improve its efficiency. An empirical comparison of the LMS algorithms is described in Section 5.

2 Chebyshev regression

In this Section we describe some properties of the Chebyshev fit and discuss its computation. Osborne and Watson (1967) proved that, if the design matrix Z of model (1) has rank p and $n > p$, then: First, there exists a Chebyshev estimate that is equal to a Chebyshev estimate for some reference set of the data; second, the Chebyshev criterion is identical to the Chebyshev criterion for the optimal reference set; third, all cases of the optimal reference set have residuals whose absolute value is equal to the Chebyshev criterion, and fourth, the rank of the design matrix of the optimal reference set is p. It is well known that the Haar condition is sufficient (but not necessary) for uniqueness of the Chebyshev estimate.

For a reference set, both the Chebyshev criterion and a Chebyshev estimate can be explicitly computed. Assume that $n = p + 1$ and the design matrix Z verifies the Haar condition. In this case two methods to obtain the Chebyshev estimate are available. The first method (see, e.g., Cheney, 1966; Meicler, 1968) is based on the LS estimate $\hat{\theta}_{LS} = (Z^tZ)^{-1}Z^tY$. Let $\hat{e}^t = (\hat{e}_1, \ldots, \hat{e}_{p+1})$ be the LS residual vector, and denote $s^t = (\mathrm{sgn}(\hat{e}_1), \ldots, \mathrm{sgn}(\hat{e}_{p+1}))$, where "sgn" represents the sign function. Then the Chebyshev criterion is

$$\omega = \begin{cases} 0 & \text{if } \sum_{i=1}^{p+1} \hat{e}_i^2 = 0 \\ \sum_{i=1}^{p+1} \hat{e}_i^2 / \sum_{i=1}^{p+1} |\hat{e}_i| & \text{otherwise} \end{cases} \tag{2}$$

and the Chebyshev estimate is

$$\hat{\theta}_C = \hat{\theta}_{LS} - \omega(Z^tZ)^{-1}Z^ts = (Z^tZ)^{-1}Z^t(Y - \omega s). \tag{3}$$

The second method (see, e.g., Meicler, 1969; Armstrong and Kung, 1980) is based on the exact fit to one of $p + 1$ possible elemental sets. The set

$\{1,\ldots,p+1\}$ is partitioned into an elemental set J and its complement. Suppose (without loss of generality) that J contains the first p indices, and let $r = p+1$ be the complementary index. Denote by Z_J and Y_J the respective submatrices of Z and Y formed with the rows indexed by J. As Z_J is nonsingular, $\hat{\theta}_J = Z_J^{-1}Y_J$ is the exact fit to the elemental set J, and $e_i = y_i - z_i^t\hat{\theta}_J$ is the ith residual based on the exact fit. Let $\xi = (\xi_1,\ldots,\xi_p) = z_r^t Z_J^{-1}$, and $\sigma^t = -\mathrm{sgn}(e_r)(\mathrm{sgn}(\xi_1),\ldots,\mathrm{sgn}(\xi_p))$. Then the Chebyshev criterion is

$$\omega = \frac{|e_r|}{1+\sum_{j=1}^{p}|\xi_i|}, \tag{4}$$

and the Chebyshev estimate is

$$\hat{\theta}_C = \hat{\theta}_J - \omega Z_J^{-1}\sigma = Z_J^{-1}(Y_J - \omega\sigma). \tag{5}$$

Remark 1 *Although for a given reference set both methods give the same ω and $\hat{\theta}_C$, an implementation based on the second method is more convenient to compute the exact LMS estimate. To our knowledge, no exact algorithm to compute the LMS estimator using the second method has yet appeared in the literature. The exact LMS algorithm of Stromberg (1993) computes the Chebyshev estimates of reference sets using the first method.*

Remark 2 *Assume $n = p+1$ and $\omega > 0$. If the matrix Z has rank p but it does not verify the Haar condition, then there exist multiple Chebyshev estimates (see, e.g., Cheney, 1966, p. 42, problems 6, 7). For our purposes, the search can be restricted to those Chebyshev estimates for which all $p+1$ residuals are equal in magnitude to ω. Using the first method, if multiple Chebyshev estimates exist, at least one $\hat{e}_i$ will be equal to zero. For these null residuals, the respective components of s may be set equal to 1 or -1, and using (3), multiple Chebyshev estimates are obtained. In the second method, an elemental set J with nonsingular design matrix has to be selected from the reference set. For this method, when multiple Chebyshev estimates exist, some ξ_i will be equal to zero. Changing conveniently the values assigned to the signs of such ξ_i's, expression (5) yields multiple Chebyshev estimates.*

When $n > p+1$, a naive algorithm computing the Chebyshev estimate examines all possible reference sets: For each reference set, it computes the Chebyshev criterion ω through (2) or (4), and selects the reference set which yields the greatest ω. Afterwards, it computes the Chebyshev estimate using (3) or (5). However, several nonexhaustive algorithms have been proposed in the literature for this purpose. These algorithms are based

on a linear programming formulation of the Chebyshev regression problem that consists of

$$\text{Minimize } \omega, \quad \text{s. t.: } -\omega \leq y_i - z_i^t\theta \leq \omega, \; i = 1, \ldots, n. \tag{6}$$

We prefer the algorithm of Armstrong and Kung (1980) (AK) which uses the dual form of (6). The source code of a FORTRAN implementation of the AK algorithm appears in Armstrong and Kung (1979). (The published code contains one harmful misprint which is corrected in Agulló, 1994). The AK algorithm obtains a finite sequence of reference sets that converges to the optimal reference set. In this sequence, consecutive reference sets only differ in one index. In each iteration, the index to be added to the current reference set would be that associated with the maximum absolute residual. The index to be dropped is selected using a rule that increases the objective function value. Notice that the AK algorithm obtains in each iteration a lower and an upper bound for the Chebyshev criterion. The lower and upper bounds are, respectively, the objective function value and the maximum absolute residual. When both bounds are identical, convergence is achieved.

3 Exhaustive LMS algorithm

Assume that matrix Z in model (1) verifies the Haar condition. Since the exact LMS estimate minimizes the hth smallest absolute residual, it must minimize the maximum absolute residual for some subset of size h of the data (i.e., $\hat{\theta}_{LMS}$ is the Chebyshev estimate to some h-subset). Moreover, from the properties of the Chebyshev regression (see Section 2), we conclude that the exact LMS estimate is equal to the Chebyshev estimate of some reference set. Further, the minimized hth absolute residual is the same as the Chebyshev criterion of this optimal reference set. Consequently, in order to compute the exact LMS estimate we can use an exhaustive algorithm that examines all reference sets. For each reference set the algorithm computes the Chebyshev estimate, and sets $\hat{\theta}_{LMS}$ as the Chebyshev estimate with smallest hth absolute residual.

It is possible to carry out some fast preliminary tests to discard those reference sets which cannot be optimal. The tests are based on the following fact. Suppose we know that the optimal hth absolute residual is not greater than ω^*. Then a reference set (with Chebyshev criterion ω and Chebyshev estimate $\hat{\theta}_C$) cannot be optimal if it verifies any of the following three

impossibility conditions:

$$\begin{array}{lrcl} [I] & \omega & > & \omega^*, \\ [II] & \#\{1 \le i \le n, \left|e_i(\hat{\theta}_C)\right| > \omega\} & > & n-h, \\ [III] & \#\{1 \le i \le n, \left|e_i(\hat{\theta}_C)\right| > \omega^*\} & > & n-h, \end{array}$$

where we use the symbol # to denote the cardinal of a set.

Stromberg's exact algorithm examines all reference sets, and computes the Chebyshev estimate $\hat{\theta}_C$ for each one using (3). It uses impossibility condition $[III]$ to avoid the computation of all absolute residuals and/or their sorting for some Chebyshev estimates. When a reference set does not verify the impossibility condition $[III]$, its Chebyshev estimate becomes the potentially best estimate and the absolute residuals are sorted to find the hth smallest absolute residual. This absolute residual becomes ω^*.

We describe next an exact LMS algorithm based also on an exhaustive enumeration of reference sets. It uses impossibility conditions $[I]$ and $[II]$ to avoid unnecessary calculations. Initially the algorithm sets $\omega^* := \infty$, and $R^* := \emptyset$. Then it considers all elemental sets $J = \{j_1, \ldots, j_p\} \subset \{1, \ldots, n\}$ with $1 \le j_1 < \ldots < j_p \le n-1$. For each elemental set J, it computes the exact fit $\hat{\theta}_J = Z_J^{-1} Y_J$. Then it examines the reference sets formed by adding an index r to J , where $j_p < r \le n$. For each reference set $R = J \cup \{r\}$, the Chebyshev criterion ω is computed using (4). If R verifies impossibility condition $[I]$, then R cannot be optimal and it is discarded. Otherwise, the algorithm computes the Chebyshev estimate $\hat{\theta}_C$ of R using (5), and calculates the Chebyshev residuals until either a) the number of absolute residuals greater than ω equals $n-h+1$, or b) the number of absolute residuals that are not greater than ω becomes equal to h. When a) occurs, R cannot be optimal (because R verifies impossibility condition $[II]$) and it is discarded. When b) occurs, the algorithm sets $\omega^* := \omega$, $R^* := R$, and $\hat{\theta}^* := \hat{\theta}_C$. When the algorithm stops, ω^* yields the optimal criterion, R^* the optimal reference set, and $\hat{\theta}^*$ the exact LMS estimate.

Matrix inversion is not needed to implement this algorithm using, for instance, an LU decomposition of Z_J. Notice that the same LU decomposition is used for evaluating, on average, $n/(p+1)$ reference sets and it is only updated when the elemental set changes.

Our algorithm differs from Stromberg's in many important respects. In Stromberg's algorithm the Chebyshev fits are based on the LS fits of reference sets. However, in our proposal the Chebyshev fits are based on exact fits of elemental sets. So, our algorithm uses the same exact fit to examine several Chebyshev fits, and it can be implemented adapting the available LMS algorithms based on elemental sets. A further benefit of our approach

is that the computation of an approximate high breakdown multivariate estimate for matrix Z (such as Rousseeuw's (1985) minimum volume ellipsoid estimate) can be done at the same time as the exact LMS computation with little additional cost (see Hawkins and Simonoff, 1993). Stromberg's algorithm requires the computation of the Chebyshev estimators for all reference sets. However, our algorithm uses impossibility condition $[I]$ to avoid the computation of the Chebyshev estimate and the Chebyshev residuals for a significant fraction of all reference sets. Moreover, when the computation of Chebyshev residuals has to be started, it seldom requires to compute all residuals and always completely avoids sorting of residuals.

The proposed algorithm examines $\binom{n}{p+1} = O(n^{p+1})$ reference sets. At its worst, it must compute n residuals for each reference set, therefore requiring at most $O(n^{p+2})$ time. On the other hand, Stromberg's algorithm requires $O(n^{p+2} \log n)$ time, but if the hth smallest absolute residual is found in $O(n)$ time, it also requires $O(n^{p+2})$ time. Notice that although both algorithms have the same computational complexity, this does not imply that their true computer times are the same, since the involved constant factors can be very different. Empirical results suggest that our proposal is about five times faster than Stromberg's algorithm (see Section 5).

Remark 3 *To analyse data sets whose design matrix has rank p and does not necessarily verify the Haar condition, the proposed algorithm requires two modifications. The first modification is concerned with the possibility of any nonsingular matrix Z_J. For each elemental set J, the numerical rank of Z_J should be checked. If the rank of Z_J is p no problem arises. When the rank of Z_J is smaller than $p-1$, the next elemental set in the list is examined. If both matrices Z_J and Z_R have rank $p-1$, the next reference set in the list is examined. Finally, if the rank of Z_J equals $p-1$ and the rank of Z_R is p, then the algorithm selects an elemental set from R whose design matrix has rank equal to p. Now, the complementary index of the elemental set plays the role of r. The second modification deals with the possibility of having some R with multiple Chebyshev estimates. The multiplicity of Chebyshev estimates occurs when some ξ_i in (4) is equal to zero. If this occurs, we must examine all Chebyshev estimates whose residuals indexed by R are equal in magnitude to ω (see Remark 2).*

4 Branch and bound algorithm

As we noted earlier, the exact LMS estimate coincides with the Chebyshev estimate of some h-subset. In fact, the optimal h-subset is the one with smallest Chebyshev criterion. In this Section we propose a Branch And

Bound (BAB) algorithm to find this optimal h-subset.

Given an index set $J_m = (j_1, \ldots, j_m) \subset \{1, \ldots, n\}$ with size m, let Z_{J_m} and Y_{J_m} be, respectively, the submatrices of Z and Y formed by the rows indexed by J_m. For the regression of Y_{J_m} on Z_{J_m}, we denote the sum of squared LS-residuals by $\phi(J_m)$, the sum of absolute values of LS-residuals by $\phi'(J_m)$, and the Chebyshev criterion by $\omega(J_m)$. It is easy to prove that:

$$J \subset J' \Longrightarrow \omega(J) \leq \omega(J'), \tag{7}$$

$$B(J_m) \leq \omega(J_m), \tag{8}$$

$$\phi(J_m) > 0 \Longrightarrow B(J_m) \leq B'(J_m) \leq \omega(J_m), \tag{9}$$

where $B(J_m) = \sqrt{\phi(J_m)/m}$ and $B'(J_m) = \phi(J_m)/\phi'(J_m)$.

Note that the monotonicity property (7) implies that the Chebyshev criterion cannot decrease if one or several cases are added to the current set of cases.

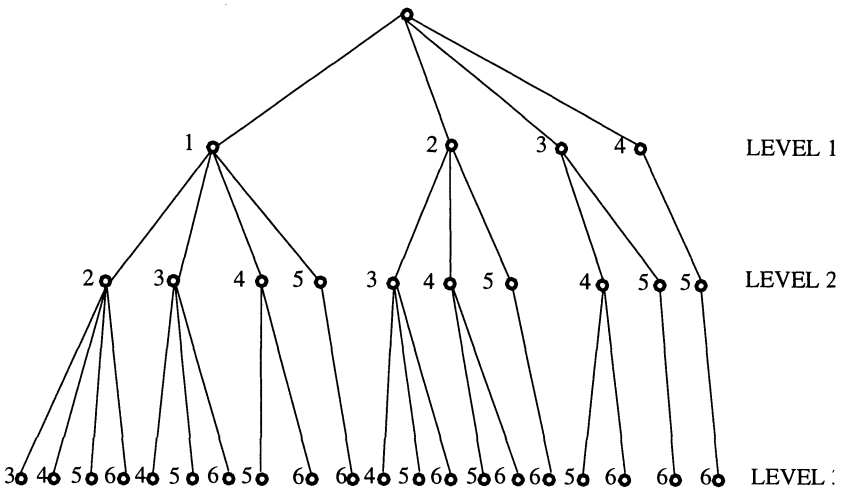
In the search of the optimal h-subset, the BAB algorithm considers subsets whose size is not greater than h. The current subset will be denoted by $J_m = (j_1, \ldots, j_m)$. We start by considering sequences that verify

$$j_1 < \ldots < j_m. \tag{10}$$

The generation of subsets is organized through a tree of nested subsets with h node levels. In each node, a further index is added from the original n indices. We use the tree described in Narendra and Fukunaga (1977). Figure 1 shows a tree for $n = 6$ and $h = 3$. A node at level m is labeled with the value of j_m and represents a subset of size m. Terminal nodes represent the $\binom{n}{h}$ possible subsets of h observations. When an exhaustive inspection of the tree is carried out, the tree is examined by moving down each branch, working from right to left. When a terminal node is reached, the inspection continues from the most recent node that has unexplored branches.

The efficiency of the BAB algorithm follows from the possibility of jumping in the exhaustive inspection sequence of the tree. In any stage of the search, let ω^* be the smallest Chebyshev criterion for a h-subset so far obtained. If the subset being considered is J_m, $1 \leq m \leq h$, and if it verifies that $\omega(J_m)$ is greater than ω^*, then, as a consequence of the monotonicity property, all h-subsets that contain J_m can be rejected implicitly.

Figure 1: Tree for $n = 6$ and $h = 3$. Labels at nodes denote the case that is added there.



We describe now the basic BAB algorithm. Initially it sets $\omega^*:=\infty$ and $J^*:=\emptyset$. If the current subset J_m verifies that $B(J_m)$ is greater than ω^*, then, from (7) and (8), the optimal h-subset cannot contain J_m, and the exploration can be continued from the most recent node that has unexplored branches. In this case, a jump occurs in the exhaustive sequence. When the current node is terminal (i.e., $m = h$) and it verifies that $B(J_h)$ is not greater than ω^*, the bound $B'(J_h)$ is computed. If $B'(J_h) \geq \omega^*$, by (7) and (9), J_h cannot be optimal, and the exploration of the tree continues. When $B'(J_h) < \omega^*$, the iterative computation of $\omega(J_h)$ (using the AK algorithm) starts. If, in some iteration of this computation, the objective function value is greater than or equal to ω^* (what implies $\omega(J_h) \geq \omega^*$), then the iterative process stops and the exploration of the tree continues. When the AK algorithm converges (i.e., $\omega(J_h)$ is smaller than ω^*), the BAB algorithm sets $\omega^*:=\omega(J_h)$ and $J^*:=J_h$, and the exploration of the tree continues. When the BAB algorithm stops, J^* is the optimal h-subset and ω^* is the minimal value of the LMS objective function.

The computation of the bound B requires the sum of squared LS-residuals. This computation is carried out by using an orthogonal decomposition procedure (Gentleman, 1974) applied to (Z_J, Y_J). When a case is added to the current subset, the orthogonal factors are updated. When the algorithm operates descending by a branch of the tree, the orthogonal factors of each level are saved. In this way, the algorithm can reselect the adequate factors when it returns to a smaller level node. This permits updating the orthogonal factors when the inspection continues by an unex-

plored branch. When a terminal node is reached and it verifies $B(J_h) < \omega^*$, then the computation of the bound $B'(J_h)$ is required. This computation is carried out quickly from the orthogonal factors. If $B'(J_h) > \omega^*$, the computation of $\omega(J_h)$ is avoided, because J_h is not optimal.

We describe now two strategies to improve the computational efficiency of the basic BAB algorithm. Suppose that $m < h$, $\text{rank}(Z_{J_m}) = p$, and $0 < B(J_m) < \omega^*$. Since the orthogonal decomposition of (Z_J, Y_J) is available, the bound $B'(J_m)$ can be quickly computed. If $B'(J_m) \geq \omega^*$, then, by (7) and (9), a jump is justified. Notice that if $m = p+1$, then $\omega(J_m) = B'(J_m)$, by (2), whereas if $m > p+1$, then $\omega(J_m) \geq B'(J_m)$. If $m > p+1$ and $B'(J_m) < \omega(J_m)$, then the iterative AK algorithm to compute $\omega(J_m)$ can be started and continued until either a) the current objective function value is greater than or equal to ω^*, or b) the current greatest absolute residual is smaller than ω^*. From the properties of the AK algorithm (see Section 2), when a) occurs, we conclude that $\omega(J_m)$ is greater than ω^*, and consequently, a jump in the exhaustive sequence is justified. However, when b) occurs, $\omega(J_m)$ is smaller than ω^*, and a jump cannot be justified.

In a node j_m at level m with ancestor nodes $j_1, \ldots, j_{m-1}$ only certain indices can be selected as successors in level $m+1$. All indices selected at nodes with labels $j_1, \ldots, j_m$ and all indices that appear in the "brother" nodes to the left of nodes $j_1, \ldots,$ and j_m are not available. Assume that the current node has n_s successors at level $m+1$. The basic BAB algorithm selects the first n_s available indices and assigns them to the successors by order from left to right in the tree. If we do not impose restriction (10) in the subset generation process, we can use a *sorting rule* for assigning available indices to the successors. Suppose that the current design matrix Z_{J_m} has rank p. For each available index i we can compute the increase in the sum of squared LS-residuals caused by adding the index i to J_m. This increase, which we call ith *partial increment*, is

$$\gamma_i = \frac{y_i - z_i^t (Z_{J_m}^t Z_{J_m})^{-1} Z_{J_m}^t Y_{J_m}}{1 + z_i^t (Z_{J_m}^t Z_{J_m})^{-1} z_i}.$$

The sorting rule selects the n_s available indices with greatest partial increments and assigns them to the successors according to the decreasing magnitude of the partial increment from left to right in the tree. If the BAB algorithm uses that sorting rule and the current subset J_m verifies that $B(J_m)$ is greater than ω^*, then the subtree emanated from the current node does not have to be examined. This follows from (7) and (8). Furthermore, if the design matrix of the previous level has rank p, as a consequence of the sorting rule all brother nodes to the left of the current node and the subtrees descending from such nodes can also be discarded.

Remark 4 *Denote* $\gamma^* = (m+1)(\omega^*)^2 - \phi(J_m)$, *and let* $\gamma' = \gamma_{n_s:n_a}$ *be the* n_s*th greatest partial increment from the* n_a *available indices. If* γ' *is greater than* γ^*, *it is not necessary to assign indices to successors, because the optimal* h*-subset cannot contain the current subset* J_m. *Sometimes the computation of partial increments for all available indices and its partial sorting can be avoided. Suppose that we proceed to compute the partial increments and maintain simultaneously a counter,* n_r, *to determine the number of partial increments that are not smaller than* γ^*. *If, at some stage of this process,* n_r *equals* n_s, *then* γ' *will necessarily be greater than* γ^*, *and a jump in the exhaustive sequence is justified.*

Even if the tree is enumerated through the sorting rule, the efficiency of the BAB algorithm depends on the initial ranking of cases in the data set. Notice that for nodes at low levels, the sorting rule assigns the first available indices to the successors. The empirical tests we have carried out suggest the convenience of reassigning the indices $1, \ldots, n$ to the cases according to the decreasing magnitude of residuals based on an approximate LMS estimate. In this way, the first h-subset evaluated by the BAB algorithm gives the approximate LMS estimate and a good upper bound for the optimal value of the objective function is found quickly. The approximate LMS estimate is computed by the BAB algorithm before starting to inspect the tree. For this task, we use a modification of the Feasible Subset Algorithm (FSA) described in Hawkins (1993).

5 Empirical comparison of the LMS algorithms

To compare the efficiency of the LMS algorithms, we have considered the following FORTRAN implementations:

- MVELMS, code of Hawkins and Simonoff (1993). We use the option that examines all elemental sets and adjusts the intercept for each elemental set.
- EXTLMS, implementation of Stromberg's exact algorithm due to Hawkins, Simonoff and Stromberg (1994).
- LULMS, our implementation of the algorithm described in Section 3. It uses an LU decomposition of the basis as described by Bartels and Golub (1969) to obtain solutions of square systems. The used code implements the first modification described in Remark 3, but not the second one.
- MVELMS1, our modification of the MVELMS code. This modification is based on the algorithm explained in Section 3, and omits from the search those elemental sets whose design matrix is singular.
- BABLMS, our implementation of the BAB algorithm described in Sec-

tion 4. It obtains the exact LMS estimate without exhaustive enumeration of h-subsets.

Table 1: Comparison of CPU times (in seconds) for LMS algorithms.

				CPU TIME (in seconds)				
DATA SET	n	p	h	EXTLMS	MVELMS1	LULMS	MVELMS	BABLMS
Aircraft	23	5	14	42.69	13.30	8.30	8.35	0.66
Coleman	20	6	13	43.13	9.95	7.09	10.44	0.32
Delivery	25	3	14	2.97	0.88	0.50	0.44	0.11
Educat	50	4	27	804.18	343.08	159.01	102.31	10.82
Hawkins	75	4	39	7774.73	4823.85	1905.11	834.07	531.31
Races	35	3	19	13.46	4.01	2.25	1.75	0.27
Salinity	28	4	16	31.92	8.85	5.38	5.16	0.43
Stacklos	21	4	12	6.48	1.65	1.21	1.15	0.11
Wood	20	6	13	43.02	9.78	7.09	10.33	0.38
Total				8762.58	5215.35	2095.94	974.00	544.90

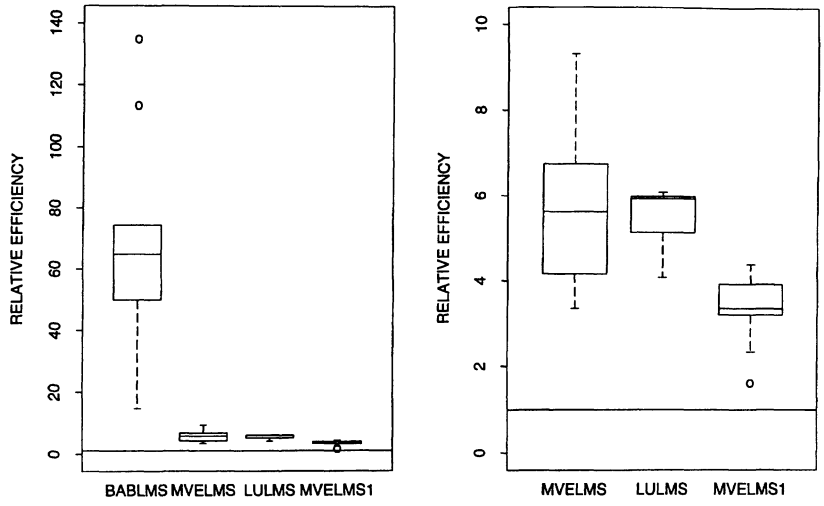
From these algorithms, MVELMS gives only an approximate LMS estimate, whereas the remaining algorithms are exact. EXTLMS, LULMS and MVELMS1 guarantee an exact LMS estimate when the observations are in general position, and BABLMS obtains a true LMS estimate when the design matrix has rank p, which is a much weaker condition. In our implementations, we use subroutines of Armstrong and Kung (1979), Miller (1992), Miller and Nguyen (1994), Ridout (1988), and Wichmann and Hill (1982). All source codes were compiled with FTN77 for 486 and were run on a 90 MHz Pentium computer. The central processor unit (CPU) times for several data sets are shown in Table 1. All data sets, excluding the one labeled "Races", can be found in Rousseeuw and Leroy (1978). The Races data set is from Atkinson (1986). Figure 2 shows the multiple boxplot of relative efficiencies, measured using the ratio of CPU time to EXTLMS CPU time.

Table 1 and Figure 2 show that for the used data sets the BABLMS algorithm dominates all other algorithms. On average, the BABLMS algorithm is about 65 times faster than the EXTLMS one, and at least one order of magnitude faster than the exhaustive MVELMS algorithm that is approximate. This may be surprising, because BABLMS searches a subset within a collection of size $\binom{n}{h}$, and this number will, in general, be much greater than, for instance, $\binom{n}{p+1}$, which is the number of subsets inspected by EXTLMS. The reason why the BABLMS algorithm is more efficient is because the number of subsets explicitly visited by it is usually smaller than the number of reference sets.

Notice that the exact LULMS algorithm is about five times faster than

the EXTLMS algorithm, and it does not need much more computer time than the approximate MVELMS algorithm based on exhaustive search over all elemental sets.

Figure 2: Multiple boxplots of relative efficiency of LMS algorithms. The baseline corresponds to the EXTLMS algorithm.



To sum up, we can conclude that for small or moderate sample sizes the proposed algorithms are more efficient than the other finite exact algorithms proposed in the literature.

References

[1] Agulló, J. (1994). A remark on algorithm AS 135. Mimeo.

[2] Armstrong, R.D., and Kung, D.S. (1979). Algorithm AS135. Min-max estimates for a linear multiple regression problem. *Appl. Statist.* **28** 93–100.

[3] Armstrong, R.D., and Kung, D. (1980). A dual method for discrete Chebychev curve fitting. *Mathematical Programming* **19** 186–199.

[4] Atkinson, A.C. (1986). Aspects of diagnostic regression analysis. *Statistical Science* **1** 397–402.

[5] Bartels, R.G., and Golub, G.H., (1969). The simplex method of linear programming using LU decomposition, *Commun. of the ACM* **12** 266–268.

[6] Cheney, E.W. (1966). *Introduction to Approximation Theory.* New York: McGraw-Hill.

[7] Gentleman, W.M. (1974). Algorithm AS75. Basic procedures for large, sparse or weighted linear least squares problems. *Appl. Statist.* **23**

448–454.

[8] Hawkins, D.M. (1993). The feasible set algorithm for least median of squares regression. *Comput. Statist. and Data Anal.* **16** 81–101.

[9] Hawkins, D.M., and Simonoff, J.S. (1993). Algorithm AS 282. High breakdown regression and multivariate estimation. *Appl. Statist.* **42** 423–441.

[10] Hawkins, D.M., Simonoff, J.S., and Stromberg, A. (1994). Distributing a computationally intensive estimator: the case of exact LMS regression. *Comput. Statist.* **9** 83–95.

[11] Meicler, M. (1968). Chebyshev solution of an inconsistent system of $n+1$ linear equations in n unknowns in terms of its least-squares solution. *SIAM Rev.* **10** 373–375.

[12] Meicler, M. (1969). A steepest ascent method for the Chebyshev problem. *Mathematics of Computation* **23** 813–817.

[13] Miller, A.J. (1992). Algorithm AS 274. *Appl. Statist.* **41** 458–478.

[14] Miller, A.J., and Nguyen, N. (1994). Algorithm AS 295. A Fedorov exchange algorithm for D-optimal design. *Appl. Statist.* **43** 669–678.

[15] Narendra, P.M., and Fukunaga, K. (1977). A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers* **26** 917–922.

[16] Osborne, M.R., and Watson, G.A. (1967). On the best linear Chebyshev approximation. *Computer Journal* **10** 172–177.

[17] Ridout, M.S. (1988). Algorithm AS 233. An improved branch and bound algorithm for feature subset selection. *Appl. Statist.* **37** 139–147.

[18] Rousseeuw, P.J. (1984). Least median of squares regression. *J. Am. Statist. Soc.* **79** 871–880.

[19] Rousseeuw, P.J. (1985). Multivariate estimation with high breakdown point. In *Mathematical Statistics and Applications, B*, Eds. W. Grossmann, G. Pflug, and W. Wertz, pp. 283-297. Dordrecht: Reidel Publishing.

[20] Rousseeuw, P.J., and Leroy, A. (1987). *Robust Regression and Outlier Detection.* New York: Wiley.

[21] Steele, J.M., and Steiger, W.L. (1986). Algorithms and complexity for LMS regression. *Discrete Applied Mathematics* **14** 93–100.

[22] Stromberg, A. (1993). Computing the exact least median of squares estimate and stability diagnostics in multiple linear regression. *SIAM J. Scient. Comput.* **14** 1289–1299.

[23] Wichman, B.A., and Hill, I.D. (1982). Algorithm AS 183. An efficient and portable pseudorandom number generator. *Appl. Statist.* **31** 188–190.