# Time- and Space-Efficient Algorithms for Least Median of Squares Regression

DIANE L. SOUVAINE and J. MICHAEL STEELE*

The least median of squared residuals regression line (or LMS line) is that line $y = ax + b$ for which the median of the residuals $|y_i - ax_i - b|^2$ is minimized over all choices of $a$ and $b$. If we rephrase the traditional ordinary least squares (OLS) problem as finding the $a$ and $b$ that minimize the mean of $|y_i - ax_i - b|^2$, one can see that in a formal sense LMS just replaces a "mean" by a "median." This way of describing LMS regression does not do justice to the remarkable properties of LMS. In fact, LMS regression behaves in ways that distinguish it greatly from OLS as well as from many other methods for robustifying OLS (see, e.g., Rousseeuw 1984). As illustrations given here show, the LMS regression line should provide a valuable tool for studying those data sets in which the usual linear model assumptions are violated by the presence of some (not too small) groups of data values that behave distinctly from the bulk of the data. This feature of LMS regression is illustrated by the fit given in Figure 1 and the residual plots of Figures 2a and 2b.

The LMS regression line is an attractive tool for data analysis, but it is not easy to compute. Steele and Steiger (1986) established that the function $f(a, b) = \text{median}\ \{|y_i - ax_i - b|^2\}$ can have on the order of $n^2$ local minima, so typical local methods have little hope of finding the global minimum of $f$. The main objective of this article is to provide algorithms that do minimize $f$ and are efficient in terms of both time and space.

Two algorithms are given here that determine the LMS regression line for $n$ points in the plane. Both of these algorithms draw their strength from the systematic use of affine duality, and one objective pursued here is the exposition of the technique of affine duality so that it will become more commonly considered by statisticians.

The first algorithm uses the so-called sweep-line technique, and it runs in worst-case time complexity $O(n^2 \log n)$, using only $O(n)$ space. The second algorithm depends on the recent development of data structures that permit the economical searching of the arrangement determined by $n$ lines. It requires only $O(n^2)$ time, but the space needed is raised to $O(n^2)$.

KEY WORDS: Duality; Sweep-line technique; Global optimization; Arrangement of lines; Arrangement searching.

## 1. THE REGRESSION PROBLEM

One aim of robust regression is to fit a line to a set of data in such a way that inspection of the residuals will detect distinctive behavior of *even a large minority* of the observations. From that point of view the solid line of Figure 1 provides a far more effective fit than does the dotted line that represents the ordinary least squares (OLS) fit. A careful analysis of the residuals from the dotted line might reveal the differences in the two clumps of data, but any glance at the residuals from the solid line will shout out that the data with positive $x$-coordinates are "special" from the perspective of the majority. This point is illustrated by the residual plots of Figures 2a and 2b.

Our main purpose here is to provide two new and efficient algorithms for calculating the robust fit that Rousseeuw (1984) called the least median of squares (LMS) regression line. This fitting process has excellent behavior from the point of view just indicated. In particular, Rousseeuw compared LMS with six important robust regression competitors and found substantial advantage to the LMS method.

To introduce the computational issues associated with LMS regression, we first let $(x_i, y_i)$ $(1 \le i \le n)$ denote $n$ points in the plane. The slope and intercept of the LMS fitted line are those values $\alpha^*$ and $\beta^*$ that minimize

$$f(\alpha, \beta) = \text{median}(|y_i - \alpha x_i - \beta|^2). \qquad (1.1)$$

As Rousseeuw showed, the LMS procedure does an excellent job following the majority. One formal method for expressing and quantifying this virtue relies on the notion of the *breakdown point* of an estimator (see, e.g., Donoho and Huber 1983). Roughly speaking, the breakdown point of an estimator is the smallest percentage of the observations that can be changed so as to make the estimate arbitrarily large. In the case of estimates of location, we can see that the breakdown point of the mean is 0%, since moving even one point out of $n$ can cause an arbitrarily large change. Similarly, one can check that the breakdown point of the median is 50%. It is also easy to see that these same figures persist in the regression case; that is, OLS has breakdown point 0% and LMS regression has breakdown point 50%.

The first mentions of an algorithm for LMS regression are in Rousseeuw (1984) and Leroy and Rousseeuw (1984). The first formal study was given in Steele and Steiger (1986), where it was shown (a) for $(x_i, y_i)$ in general position, $f(\alpha, \beta)$ has $\Omega(n^2)$ local minima ($\Omega$ stands for "order of at least" just as $O$ stands for "order of at most," and for this result general position means that no three points are colinear and no four points determine parallel lines), and (b) exact minimization of $f$ is reducible to a finite search that has worst time complexity of $O(n^3)$.

Although an $O(n^3)$ algorithm is practical on modest data sets, faster algorithms are needed if the LMS fitting procedure is to be useful as an interactive tool on data sets of even moderate size. The main algorithm we give here runs in time $O(n^2)$. Because lower bounds for geometric algorithms are extremely rare (and generally very crude), it is to be expected that establishing the optimality of the $O(n^2)$ result will be difficult. Still, in all likelihood, there is no algorithm for LMS regression that is faster than $O(n^2)$, and applications of LMS regression may have to live with

* Diane L. Souvaine is Assistant Professor, Department of Computer Science, Rutgers University, New Brunswick, NJ 08903. J. Michael Steele is Professor, Department of Civil Engineering and Operations Research, Program in Engineering Statistics, Princeton University, Princeton, NJ 08544. Souvaine's research was supported in part by National Science Foundation Grant MCS83-03926. Steele's research was supported in part by National Science Foundation Grant DMS-8414069.

© 1987 American Statistical Association
Journal of the American Statistical Association
September 1987, Vol. 82, No. 399, Theory and Methods

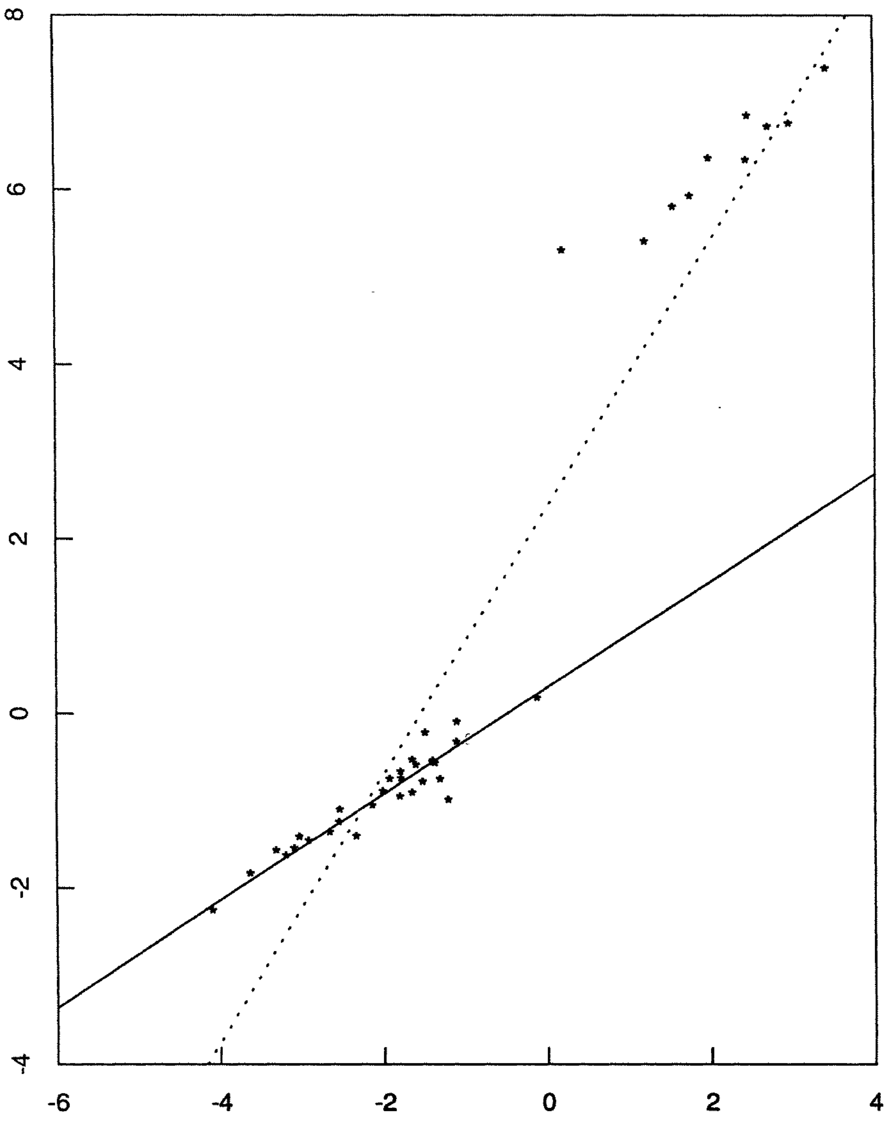


*Figure 1. Benefits of High Breakdown Regression.*

that constraint. One should note that an approximate algorithm such as that of Leroy and Rousseeuw (1984) may give useful fits at a computational cost that may be substantially smaller than $O(n^2)$. Such approximate algorithms may be the only practical approach for LMS multiple regression.

We also give a second algorithm in which more careful attention is paid to the space requirements of LMS regression. Although our $O(n^2)$ algorithm requires $O(n^2)$ space, we are able to give a $O(n^2 \log n)$ algorithm that requires only $O(n)$ space. There are certainly machines and problems for which the second algorithm would actually run faster because of the lessened overhead of memory management.

The algorithms we provide use two important new tools from computational geometry. These tools are (a) affine duality and (b) data structures that facilitate searching arrangements. The benefit of this feature is that these tools work together very well and should prove useful in many other statistical problems that deal with lines and points.

The algorithms given here have been tailored to be as simple as possible given the aim of attaining the achieved orders of space and time complexity. Obviously any implementation of these algorithms would entail numerous minor (or not so minor) speedups and improvements; in particular, we could have been much more stingy with our pointers. This situation rests in comfort with the widely endorsed philosophy of getting the right algorithm and then toning for high performance. One can consult Bentley (1982) for many elegant illustrations of this viewpoint.

*Technical Remark.* The word *median* needs to be unambiguously defined in the case of even $n$. Our convention for $n$ even will be to take the *high median* or the value of rank $m = 1 + n/2$ in an ordered list $u_1 \leq u_2 \leq \cdots \leq u_n$. This convention is made for specificity and clarity; it has no material impact on the analysis of our algorithms.

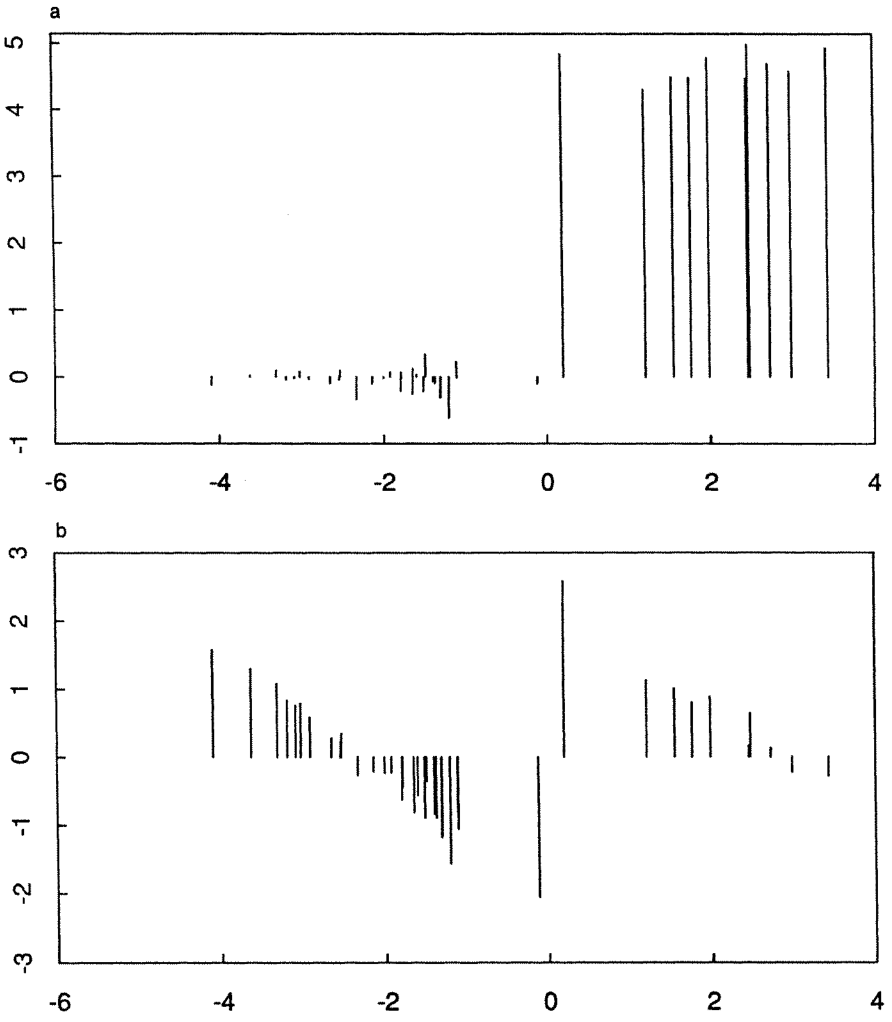


*Figure 2. (a) Residuals From Robust Line. (b) Residuals From OLS's Line.*

## 2. GEOMETRIC TRANSFORMATIONS: POINT/LINE DUALITY

There have been powerful applications of transformations in geometry since the work of Steiner and Poncelet, but the thesis of Brown (1979) opened up the floodgate of applications to computational geometry (see, e.g., Chazelle, Guibas, and Lee 1983; Dobkin and Souvaine 1986; Edelsbrunner, O'Rourke, and Seidel 1983).

In statistical work the point/line duality was used most recently in Johnstone and Velleman (1985), which gives an algorithm for a different robust regression method, the resistant line. In data analysis and in simple linear regression, duality was also used by Emerson and Hoaglin (1983), Dolby (1960), and Daniels (1954).

The transformation that we will apply is the mapping $T$, which takes the point $(a, b)$ to the line $y = ax + b$ and takes the line $y = ax + b$ to the point $(-a, b)$. The minus sign in $(-a, b)$ may be a little surprising, but it is precisely what is required to make $T$ a bona fide duality.

Under this transformation, the point $P$ that is determined by two lines $L_1$ and $L_2$ is mapped to the line $TP$, which is determined by the two points $TL_1$ and $TL_2$. Likewise the line $L$ determined by two points $P_1$ and $P_2$ is mapped to the point $TL$ determined by the two lines $TP_1$ and $TP_2$. These relations lie at the base of duality, and they are shared by the classical transformation $S$ of Poncelet, which maps the point $(a, b)$ to the line $ax + by + 1 = 0$ and maps the line $ax + by + 1 = 0$ to the point $(a, b)$. The duality applied here is not as symmetrical as Poncelet's; in particular, we note $S^2 = I$, but since $y = ax + b$ goes to $(-a, b)$ we have $T^2 \neq I$.

The duality given by $S$ may have more algebraic appeal, but $T$ has additional ordering properties that make it more useful in computational problems where order plays a role. The first important ordering property is that the transformation $T$ preserves the relationship of "above" and "below" for points and lines. That is, if $P$ is above $L$ then $TP$ is above $TL$. The transformation $T$ also preserves the vertical distance between points and lines; $TL$ is above $TP$ by exactly the same distance that $L$ is above $P$, and so on. This property is naturally crucial in a problem where a median residual is to be minimized over a selection of lines.

The second important invariance property of $T$ is that if the slope of $L$ is larger than the slope of $L'$, then the $x$-coordinate of $TL$ is smaller than the $x$-coordinate of $TL'$. This is trivial from the definition, but is a very nice relation in practical applications.

We should make one final distinction about the way we will use duality and the transformation $T$. We will make no use of the so-called *ideal* objects, like "the point at infinity," which help give the projective plane much of its charm. All of our computations depend on ordinary objects in the finite plane.

## 3. EQUIOSCILLATION AND ITS DUAL

To articulate a finite problem that can be precisely dualized, it is worthwhile to introduce some terminology relevant to the local structure of LMS regressions. First for any $\alpha$ and $\beta$ the line $l_{\alpha,\beta} = \{(x, y): y = \alpha x + \beta\}$ defines residuals $r_i(\alpha, \beta)$, which we can typically write as $r_i$ without fear of confusion. We say the line $l_{\alpha,\beta}$ *bisects* three distinct points $(x_{i_j}, y_{i_j})$ $(j = 1, 2, 3)$ if all of the $r_{i_j}$ are of the same magnitude $r$ but not all have the same sign. If $x_{i_1} < x_{i_2} < x_{i_3}$ and $r_{i_1} = -r_{i_2} = r_{i_3}$ we say $l_{\alpha,\beta}$ *equioscillates* with respect to the points.

It was proved in Steele and Steiger (1986) that the LMS regression line must be an equioscillating line relative to some triple of data points. Since there are at most $\binom{n}{3}$ such lines a naive algorithm would be to examine them all. Obviously our $O(n^2)$ algorithm must work differently.

Given an equioscillating line $l_{\alpha,\beta}$ there are two related lines with slope $\alpha$; the line $L_1$ determined by the points $P_1 = (x_{i_1}, y_{i_1})$ and $P_3 = (x_{i_3}, y_{i_3})$, and the line $L_2$ that goes through the point $P_2 = (x_{i_2}, y_{i_2})$. A key property of the LMS regression line $l_{\alpha,\beta}$ is that the number $K$ of points between $L_1$ and $L_2$ must satisfy

$$\begin{aligned} K &= (n - 4)/2 \quad n \text{ even} \\ &= (n - 5)/2 \quad n \text{ odd} \end{aligned} \tag{3.1}$$

provided $n > 3$ (compare Steele and Steiger 1986, main lemma). Both the idea of equioscillation and the enclosure property given by Equation (3.1) are illustrated in Figure 3.

One simple method of determining the LMS regression line can now be given. For each triple of data points $P_1$, $P_2$, and $P_3$ that are ordered by $x$-coordinate we first determine lines $L_1$ and $L_2$, as before. Next we find a triple with exactly $K$ data points between $L_1$ and $L_2$ such that the vertical distance between $L_1$ and $L_2$ is minimized. The LMS regression line is the line that goes exactly between $L_1$ and $L_2$.

We can now see the rephrasing of our problem obtained by applying $T$ to the data points and the lines they determine. The condition that there are $K$ points between the lines $L_1$ and $L_2$ becomes the condition that there are $K$ lines between the points $TL_1$ and $TL_2$. Further, $TL_1$ is the point determined by the two lines $TP_1$ and $TP_3$; $TL_2$ is the point on the line $TP_2$ that has the same $x$-coordinate as the point $TL_1$.

Our algorithmic problem in the dual can now be ex-

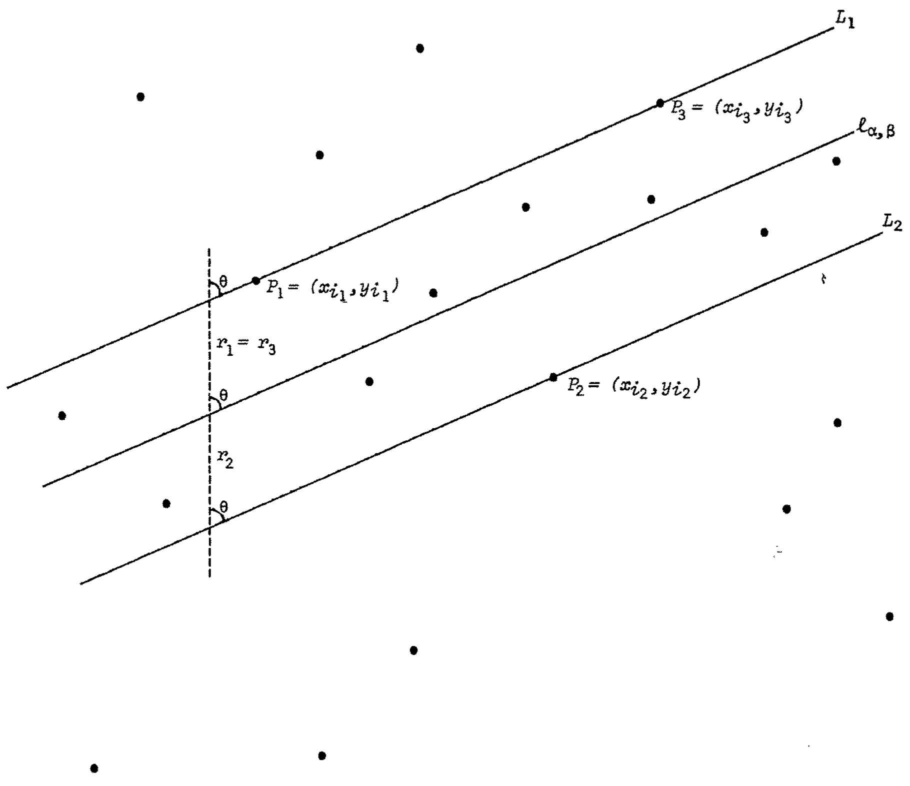

$L_1$
$P_3 = (x_{i_3}, y_{i_3})$
$l_{\alpha,\beta}$
$L_2$
$\theta$
$P_1 = (x_{i_1}, y_{i_1})$
$r_1 = r_3$
$\theta$
$P_2 = (x_{i_2}, y_{i_2})$
$r_2$
$\theta$

*Figure 3. Equioscillation. For the given set of data points of size $n = 20$, the LMS regression line $l_{\alpha,\beta}$ equioscillates relative to $P_1$, $P_2$, and $P_3$. Note that $L_1 \| l_{\alpha,\beta} \| L_2$ and lines are equally spaced.*

pressed as follows: Given $n$ lines $L_i$ ($1 \le i \le n$) in general position in the plane with intersection points $P_{ij}$ ($1 \le i < j \le n$), find that line $L^*$ and intersection point $P^*$ such that, among all line–point pairs $(L, P)$ that have exactly $K$ of the $L_i$ cutting the vertical segment $S$ joining $L$ and $P$, the pair $(L^*, P^*)$ has the smallest vertical distance. [For statistical purposes there is no loss in assuming that our $n$ lines are in general position in the sense that no pair of lines are parallel and no more than two meet at any given point. If this were not the case a statistically subliminal perturbation of the $L_i$ could make it so. This perturbation (if required) can be achieved with a preprocessing cost of order $O(n)$. In addition, with minor adjustments to our algorithms, one can remove the stipulation that the lines be in general position.]

If this *dual problem* is solved, then in constant time we can interpret the solution as a solution to the primal.

## 4. SWEEP-LINE ALGORITHM AND SPACE EFFICIENCY

We are now in a position to give our first duality-based algorithm for computing the least median of residuals regression line. We will suppose for this section that the duality transformation has been performed, and we proceed to solve the dual problem spelled out previously.

The technique that we apply is a type of sweeping of the plane by combinatorially moving a vertical "sweep-line" from the left to the right, stopping at each of the $P_{ij}$ to perform some computation. This is probably the first application of these techniques in the area of regression, but Shamos and Hoey (1976) and Bentley and Ottman (1979) showed that sweep-line techniques are applicable to a variety of problems in computational geometry. Edelsbrunner and Wetzl (1986) recently applied techniques similar to those presented here to achieve fast algorithms for such problems as half-planar range estimation, the $k$-nearest neighbors search problem, and minimum area triangulation.

The computational objectives of this first algorithm are (a) to illustrate the power of duality in a statistical problem and (b) to provide a reasonably fast algorithm that is optimally space efficient. The second objective requires some care and a few subtleties. In the first place, by "space-efficient" we mean an algorithm that requires $O(n)$ space. Since the problem is of this size one cannot do better.

The sweep-line algorithm requires two off-the-shelf data structures. The first of these, to be called LIST, will maintain our set of $n$ lines in a geometrically meaningful order. LIST can be implemented as a simple linked list, but we will want to add some additional pointers that will help LIST to interact with the second data structure.

Our second structure will be used to store a certain subset of $n - 1$ out of the set of $n(n - 1)/2$ intersection points $P_{ij}$. The structure we create will permit us to access the "smallest" element in time $O(1)$ and permits the insertion or deletion of an element in time $O(\log n)$. The ordering we will use to give meaning to "smallest" is that $P_{ij} \ll P_{st}$ provided the $x$-coordinate of $P_{ij}$ is smaller than the $x$-coordinate of $P_{st}$. To implement this second structure we can use a heap (see, e.g., Aho, Hopcroft, and Ullman 1974). We will refer to our particular structure as HEAP.

The initialization of these structures will show how they will be used and will help make the algorithm transparent. We begin by noting that to the left of all of the intersection points, the lines are themselves ordered by slope, that is, the line with smallest slope is above all of the other lines, and so on. Consequently, we place all of the $L_i$ into LIST in increasing order of slope. We also augment the structure holding the $L_i$ by storing along with each of the $L_i$ a set of four pointers, Up1, Up$K$, Down1, and Down$K$, which will point to the lines that are 1 above $L_i$, $K$ above $L_i$, and so forth in the ordering given by LIST. Since lines too near the top or the bottom will not have lines satisfying the required conditions, we set the pointers to the null value to flag this situation.

The current order of the lines in LIST, increasing order of slope, is exactly equivalent to decreasing order of $L_o$-intercept, where $L_o$: $x = A$ is any vertical line strictly to the left of all of the $P_{ij}$. To compute a potential value for $A$, we must locate the leftmost intersection point $P_{ij}$. But the leftmost $P_{ij}$ must be an intersection of two lines adjacent in LIST. Consequently, we pass through LIST and for each adjacent pair $L_i$ and $L_j$ in LIST, we install $P_{ij}$ in HEAP. We do this while respecting the ordering of the $P_{ij}$ by $x$-coordinates and we end up with a heap of size $n - 1$ with the leftmost of the $P_{ij}$ installed at the root. As an important adjunct to the heap building process, we make a double set of pointers that associate the elements of HEAP and the elements of LIST. Specifically, for each adjacent pair $L_i$ and $L_j$ in LIST and each corresponding intersection point in HEAP, we create pointers from $L_i$ and $L_j$ to $P_{ij}$ and pointers from $P_{ij}$ to $L_i$ and $L_j$.

Once HEAP has been constructed, we fix the line $L_o$: $x = A$ by setting $A$ to some arbitrary value less than the $x$-coordinate of the point at the root of HEAP. Next we introduce another vertical line $L$, which we call the sweep-line, and we initialize $L$ to $L_o$. The line $L$ is not essential to the algorithm, but it helps articulate a program invariant that makes verifications clearer.

With the preprocessing and initialization phase complete, we should make a preliminary tally of resources consumed. First, because of the sorting according to slope, LIST requires time $O(n \log n)$ to build and space $O(n)$ to store. Since HEAP is also of size only $O(n)$, it can be created in time $O(n)$ and space $O(n)$ using well-known methods.

The algorithm for computing the LMS regression line is now almost trivial using the data structures we have built. The picture to keep in mind is that of sweeping the vertical line $L$ from $L_o$ until it has passed through all of the points. What we will do is manage LIST and HEAP in such a way that (a) LIST always provides the ordering of the $L_i$ according to decreasing order of $L$-intercept, (b) the pointers stored in LIST remain consistent with their initial definition, and (c) HEAP always stores as its minimal element the intersection point $P_{ij}$ that has smallest $x$-coordinate of any intersection point to the right of $L$.

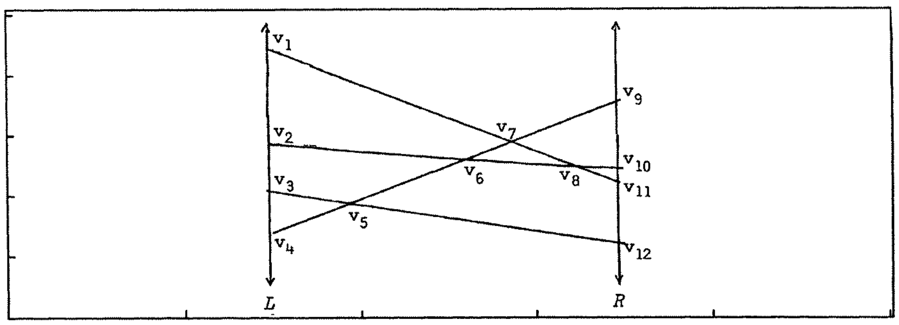
As $L$ reaches each new intersection point $P_{ij}$ we compute the vertical distance to the lines Up$K$ and Down$K$ and compare the results with the previous optimum. Then we exchange the relative positions of $L_i$ and $L_j$ in LIST. A key geometric observation parallel to the idea used to compute $L_o$ is that from the whole set of $n(n-1)/2$ intersection points $P_{ij}$ $(1 \le i < j \le n)$ the one that is nearest to the sweep-line $L$ must be determined by a pair of lines $\{L_i, L_j\}$, which are adjacent in the updated LIST. This observation permits us to keep a set of candidates for the "next point to explore," which is at most of cardinality $n - 1$ and means that for each of the $O(n^2)$ new positions of $L$, we need update only three points of HEAP at a cost of $O(\log n)$ time each. Thus the entire algorithm runs in time $O(n^2 \log n)$.

## 5. DATA STRUCTURES FOR SEARCHING ARRANGEMENTS

To improve upon the $O(n^2 \log n)$ time complexity of our first algorithm, we introduce the idea of a *hammock* (Chazelle 1984). A hammock is a graph $G$ constructed from the $n$ given lines $L_i$ $(1 \le i \le n)$ and an auxiliary pair of vertical lines $L$ and $R$, called "left" and "right." If $m$ is the number of intersections among the $L_i$ occurring between the two lines $L$ and $R$, the hammock has a vertex set $V$ of cardinality $m + 2n$ formed by adding the set of intersection points on the vertical lines to the set of intersection points enclosed by $L$ and $R$. The edge set $E$ of $G$ consists of those pairs of $V$ that are joined by a *segment of the arrangement*. That is, $v_i$ and $v_j$ are joined if they are on the same line and there is no line that separates them. An illustration of a hammock is given in Figure 4a.

The key computational fact we need about hammocks is that Chazelle (1984) introduced an algorithm that computes either the edge list representation or the adjacency list representation of a hammock in time $O(n \log n + m)$ and space $O(n + m)$. For our application, we must pick $L$ and $R$ so that the hammock contains all of the intersection points $P_{ij}$ $(1 \le i < j \le n)$. Assuming as before that all $n$ lines are in general position, we may conclude that $m = n(n-1)/2$ and thus that the hammock will require

a



b

1: $v_1 \to v_2 \to v_7$
2: $v_2 \to v_1 \to v_3 \to v_6$
3: $v_3 \to v_2 \to v_4 \to v_5$
4: $v_4 \to v_3 \to v_5$
5: $v_5 \to v_3 \to v_4 \to v_{12} \to v_6$
6: $v_6 \to v_2 \to v_5 \to v_8 \to v_7$
7: $v_7 \to v_1 \to v_6 \to v_8 \to v_9$
8: $v_8 \to v_7 \to v_6 \to v_{12} \to v_{10}$
9: $v_9 \to v_7 \to v_{10}$
10: $v_{10} \to v_8 \to v_9 \to v_{11}$
11: $v_{11} \to v_8 \to v_{10} \to v_{12}$
12: $v_{12} \to v_5 \to v_{11}$

c

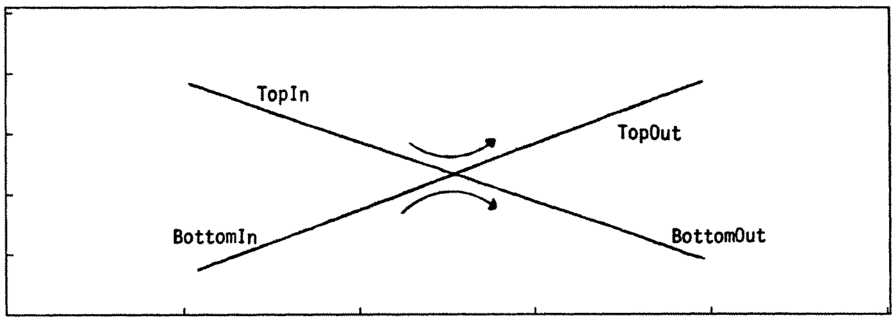


*Figure 4. (a) Small Hammock. (b) Adjacency List for the Hammock. (c) Entrance/Exit Rules.*

$O(n^2)$ time and space. Furthermore, all of the vertices of $G$ that are on $L$ or $R$ have degree 2 or 3 and all other vertices have degree 4.

The adjacency list representation of $G$, our choice, contains a linked list of the vertices such that each vertex $v$ also has a pointer to a linked list containing the vertices adjacent to $v$ in counterclockwise order. An illustration of the adjacency list representation corresponding to the hammock of Figure 4a is given in Figure 4b. In this bounded degree situation, the linked adjacency list can be replaced with just a simple four-element array. We shall tag the four edges associated with a vertex not on $L$ or $R$ with labels like those in Figure 4c.

A final point about Chazelle's $O(n^2)$ construction of a complete hammock is that it is reasonably easy to implement and there is not a substantial amount of overhead being concealed in the constants of the $O(n^2)$ bound. A contrasting example is Strassen's $O(n^{2.81})$ algorithm for fast matrix multiplication, which has seen little practical implementation despite its remarkable theoretical importance (see, e.g., Aho et al. 1974).

## 6. TIME-EFFICIENT LMS REGRESSIONS

In addition to the hammock just described we will need one further data structure before we can detail our second algorithm. Just to have a name we will call this additional structure $S$. If $Q_i$ ($1 \le i \le n$) are the points of intersection of $L_i$ on $L$, we require that (a) $S$ be a linked list of the $L_i$ ordered by decreasing $y$-coordinate of $Q_i$, (b) for each $L_i$ we have defined a pointer called $K$below($L_i$), which points to the line $L_j$ that is $K + 1$ below $L_i$ in the list $S$, and (c) we have defined a similar pointer $K$above($L_i$), which points to a line $L_k$ that is $K + 1$ above $L_i$ in $S$. If one of the lines $L_j$ or $L_k$ fails to exist, it is understood that the corresponding pointer takes on the null value.

A notion that helps the understanding of our second procedure is that of a *strand*. By a strand we mean that connected sequence of intervals that is obtained by starting at a point $Q_i$ following the only non-$L$ edge of the arrangement from $Q_i$ and then successively following the traversal rule given in Figure 4c and making the unique permissible transition from each of the $P_{ij}$ until we arrive at $R$. The key property of a pair of strands $W$ and $W'$ is that any vertical line between two points of $W$ and $W'$ will always cross exactly the same number of lines $L_i$ ($1 \le i \le n$).

The algorithm is now easy. For each $Q_i$ ($1 \le i \le n$) we construct the strand $W_i$ that originates from $Q_i$. Using the pointer in $S$ we find the $Q_j$ and $Q_k$ that are $K + 1$ above or $K + 1$ below $Q_i$ (if they exist). We then consider the associated strands $W_j$ and $W_k$.

For specificity let us suppose that $W_j$ is a strand that is $K + 1$ below $W_i$. We will traverse $W_i$ by starting at $Q_i$ and taking the unique admissible edge out of each successive intersection point until we reach $R$. At each step of this process we increment our position in $W_j$ until we locate the edge $E$ of $W_j$ that lies directly below our current intersection point $P$ in $W_i$. We then compute the vertical distance between $P$ and $E$. As we traverse $W_i$ we retain a record of the minimum value found and the point–edge pair that produces that minimum.

Although the details have just been spelled out for $W_i$ and the strand $K + 1$ below $W_i$, we obviously can process the strand that is $K + 1$ above $W_i$ in the same way. (Typically one has either a strand $K + 1$ above or a strand $K + 1$ below, but it is also possible to have both.)

After each strand $W_i$ has been processed we can then at cost $O(n)$ examine the minimal point–edge pairs associated with the $W_i$ ($1 \le i \le n$). This will permit us to find the global minimum, that is, the point–edge pair ($P_o$, $E_o$) such that the vertical distance between $P_o$ and $E_o$ is minimal in the class of all point–edge pairs $P$ and $E$ such that $K$ lines pass between $P$ and $E$. This pair ($P_o$, $E_o$) is (of course) a solution to our dual version of the LMS regression problem.

Before leaving this algorithm we should note where resources are consumed. The creation of the basic hammock had time cost $O(n^2)$ and space cost $O(n^2)$. Creating $S$ added the more modest space cost of $O(n)$ and time cost $O(n \log n)$. There were $n$ strands traversed, and each of these traversals required time $O(n)$, so the cumulative time cost of the core algorithm was $O(n^2)$. The resources consumed over the whole process are, therefore, of order $O(n^2)$ for both space and time.

## 7. CONCLUDING REMARKS

The most obvious question that we have not fully addressed is that of multiple regression. The techniques of this article should extend, although the expositional and analytical overhead increases substantially. The fundamental tool for searching a higher-dimensional arrangement has been provided in Edelsbrunner et al. (1983), and the basic device of duality remains intact. The running time of any exact LMS regression with $d$ dimensional is likely to be no faster than $O(n^d)$. This fact makes the study of heuristic alternatives to LMS regression an open and interesting area for further study.

Two other issues of importance are (a) making these algorithms dynamic in the sense of being able to add or delete observations cheaply and maintain an LMS regression line, and (b) providing analogies to LMS regression that accommodate weights for different data points.

The importance of these issues comes from the possibility of real time applications of robust regression as part of filtering systems.

[*Received March 1986. Revised December 1986.*]

## REFERENCES

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1974), *The Design and Analysis of Computer Algorithms,* Reading, MA: Addison-Wesley.

Bentley, J. L. (1982), *Writing Efficient Programs,* Prentice-Hall Software Series, Englewood Cliffs, NJ: Prentice-Hall.

Bentley, J. L., and Ottman, T. A. (1979), "Algorithms for Reporting and Counting Geometric Intersections," *IEEE Transactions on Computers,* C-28, 643–647.

Brown, K. Q. (1979), "Geometric Transforms for Fast Geometric Algorithms," Technical Report CMU-CS-80-101, Carnegie-Mellon University, Dept. of Computer Science.

Chazelle, B. (1984), "Intersecting Is Easier Than Sorting," in *Proceed-*

*ings of the 16th Association for Computing Machinery, Symposium on the Theory of Computing,* pp. 125–135.

Chazelle, B., Guibas, L. J., and Lee, D. T. (1983), "The Power of Geometric Duality," in *Proceedings of the 24th IEEE Foundations of Computer Science,* pp. 217–225.

Daniels, H. E. (1954), "A Distribution-Free Test for Regression Parameters," *Annals of Mathematical Statistics,* 25, 499–513.

Dobkin, D. P., and Souvaine, D. L. (1986), "Computational Geometry—A Users Guide," in *Advances in Robotics I: Algorithmic and Geometric Aspects of Robotics,* eds. J. T. Schwartz and C. K. Yap, Hillsdale, NJ: Lawrence Erlbaum.

Dolby, J. L. (1960), "Graphical Procedures for Fitting the Best Line to a Set of Points," *Technometrics,* 2, 477–481.

Donoho, D. L., and Huber, P. J. (1983), "The Notion of Breakdown Point," in *A Festschrift for Erich L. Lehmann,* eds. P. J. Birkel, K. Doksum, and J. L. Hodges, Belmont, CA: Wadsworth, pp. 157–184.

Edelsbrunner, H., O'Rourke, J., and Seidel, R. (1983), "Constructing Arrangements of Lines and Hyperplanes With Applications," in *Proceedings of the 24th IEEE Foundations of Computer Science,* pp. 83–91.

Edelsbrunner, H., and Wetzl, E. (1986), "Constructing Belts in Two-Dimensional Arrangements With Applications," *SIAM Journal on Computing,* 15, 271–284.

Emerson, J. D., and Hoaglin, D. C. (1983), "Resistant Lines for $y$-versus-$x$," in *Understanding Robust and Exploratory Data Analysis,* eds. D. C. Hoaglin, F. Mosteller, and J. Tukey, New York: John Wiley.

Johnstone, I. M., and Velleman, P. F. (1985), "The Resistant Line and Related Regression Methods," *Journal of the American Statistical Association,* 80, 1041–1054.

Leroy, A., and Rousseeuw, P. J. (1984), "PROGRESS: A Program for Robust Regression," Report 201, Centrum voor Statistiek en Operationell Onderzoek, University of Brussels.

Rousseeuw, P. J. (1984), "Least Median of Squares Regression," *Journal of the American Statistical Association,* 79, 871–880.

Shamos, M., and Hoey, D. (1976), "Geometric Intersection Problems," in *Proceedings of the 17th IEEE Foundations of Computer Science,* pp. 208–215.

Steele, J. M., and Steiger, W. L. (1986), "Algorithms and Complexity for Least Median of Squares Regression," *Discrete Applied Mathematics,* 13, 509–517.