Univariate Adaptive Thinning

Nira Dyn, Michael S. Floater, and Armin Iske

Abstract. In this paper we approximate large sets of univariate data by piecewise linear functions which interpolate subsets of the data, using adaptive thinning strategies. Rather than minimize the global error at each removal (AT0), we propose a much cheaper thinning strategy (AT1) which only minimizes errors locally. Interestingly, the two strategies are equivalent in all our numerical tests and we prove this to be true for convex data. We also compare with non-adaptive thinning strategies.

§1. Introduction

In applications such as visualization, it is often desirable to generate a hierarchy of coarser and coarser representations of a given discrete data set. Though we are primarily interested in hierarchies of scattered data sets, and in particular piecewise linear approximations over triangulations in the plane [1], we focus in this paper on univariate data sets and propose several adaptive thinning strategies. Thinning algorithms generate hierarchies of subsets by removing points from the given data set one by one, in such a way that the 'least' significant point is removed at each step, according to some desirable criterion. Our criterion here will primarily be the minimization of approximation error, so our thinning algorithms are *adaptive*. This is in contrast, for example, to the thinning strategies of [2,3], where the criterion was to generate subsets of well distributed points, independent of the height values.

Thinning algorithms for piecewise linear approximation to univariate data have appeared before in the literature as *decimation* algorithms, as in Heckbert and Garland [4], and as *knot removal* for linear splines, as in Lyche [6].

In this paper, we design, test, and compare four methods for anticipating the error incurred by the removal of a point from the current subset. Our algorithms choose the point to be removed as the one of minimal anticipated error. Our main conclusion is that the algorithm AT1, which is based on making a local error estimate, but taking account of all previously removed points, is the best algorithm from the point of view of our numerical results

Mathematical Methods for Curves and Surfaces: Oslo 2000
Tom Lyche and Larry L. Schumaker (eds.), pp. 123–134.

ISBN 0-8265-1378-6

and theoretical analysis. In fact our theoretical analysis shows that its computational complexity is $O(N \log N)$, with N the number of points in the data set, provided one uses a heap to store the anticipated errors. Moreover, we prove that for data sampled from a convex function, AT1 minimizes the global approximation error at every step. These latter two results extend to piecewise linear functions over triangulations for scattered data in the plane; see [1].

§2. Adaptive Thinning

Suppose $[a, b]$ is a real interval and that $X = (x_1, \dots, x_N)$ is a given sequence of points in $[a, b]$ such that

$$a = x_1 < x_2 < \cdots < x_N = b.$$

Suppose further that some unknown function $f : [a, b] \to \mathbb{R}$ is sampled at these points, giving the values $f(x_1), \dots, f(x_N)$.

For each n, $1 < n < N$, we are interested in finding a subset $Y = (y_1, \dots, y_n)$ of X, such that

$$a = x_1 = y_1 < y_2 < \cdots < y_n = x_N = b, \tag{1}$$

and such that the piecewise linear interpolant $L(f, Y)$ to the data

$$\{(y, f(y)) : y \in Y\}$$

is close to the given data $\{(x, f(x)) : x \in X\}$, in the sense that the error

$$E(Y; X; f) = \max_{x \in X} |L(f, Y)(x) - f(x)| \tag{2}$$

is small relative to the errors corresponding to other subsets of X of cardinality n. To guarantee that $E(Y; X; f)$ is well defined, we refrain from removing the points x_1, x_N, so that $L(f, Y)$ is defined on $[a, b]$.

Ideally, for any given n, $1 < n < N$, we would like to find a subset Y of X of cardinality n for which the error in (2) is minimal. However, it is clearly impractical to search amongst all possible subsets, and this motivates the more pragmatic approach of thinning.

The idea of thinning is to remove points from X one by one in order to reach a subset Y of a certain size. In general we want to remove a point of 'least' significance. Our criterion for removing a point from the current subset is to minimize its *anticipated error*, which is an estimate of the error incurred by the removal of the point with respect to some error measure. Thus the thinning algorithm is a greedy algorithm, choosing the current step to do the optimal step in the current situation.

We define our thinning algorithm by saying that a point y_i in Y, $1 < i < n$, is removable if

$$e(y_i) = \min_{j=2,3,\dots,n-1} e(y_j), \tag{3}$$

where $e(\cdot)$ is our chosen anticipated error.

Thinning Algorithm

1) Set $X_N = X$.
2) For $i = N, N-1, \ldots, 3$:

   locate a removable point x in X_i and set $X_{i-1} = X_i \setminus x$.

The result of the thinning algorithm is a hierarchical sequence of subsets of X,

$$\{a, b\} = X_2 \subset X_3 \subset \cdots \subset X_N = X,$$

with $|X_i| = i$.

Now we consider various anticipated error measures $e(\cdot)$, each of which defines a removable point in (3), and results in a different algorithm. The algorithm is termed Adaptive Thinning whenever the anticipated error depends on some of the function values $\{f(x) : x \in X\}$.

Algorithm AT0

In this algorithm the anticipated error of a point y_i is the maximum of the errors incurred by the removal of y_i at all the points of X,

$$e_0(y_i; Y) = E(Y \setminus y_i; X; f) = \max_{x \in X} |L(f, Y \setminus y_i)(x) - f(x)|. \tag{4}$$

Indeed, $e_0(y_i)$ is the actual error incurred by the removal of y_i, measured in the sup-norm over X.

Algorithm AT1

A less expensive to compute measure of anticipated error is

$$e_1(y_i; Y) = e_{[y_{i-1}, y_{i+1}]}, \tag{5}$$

where for any interval I whose endpoints I_B belong to X, e_I is defined as

$$e_I = \max_{x \in I \cap X} |L(f, I_B)(x) - f(x)|.$$

Note that here we consider only the error incurred by the removal of a point at those points of X which belong to the current interval of the removed point.

Algorithm AT2

In this algorithm the anticipated error is similar to the one in AT1, but does not depend on the points that are already removed. This anticipated error is simpler to compute than the anticipated error of AT1,

$$e_2(y_i; Y) = |L(f, \{y_{i-1}, y_{i+1}\})(y_i) - f(y_i)|. \tag{6}$$

Algorithm NAT

Here the removal of a point depends only on the density of the points Y, and is independent of f. Thus it is a Non-Adaptive Thinning algorithm. In fact $e(\cdot)$ is such that the removal of points results in approximately equidistributed sets of points X_i, for intermediate values of i,

$$e_3(y_i;Y) = (y_i - y_{i-1})(y_{i+1} - y_i). \tag{7}$$

When the set Y is fixed we use also the notation $e_j(y) = e_j(y;Y)$ for $j = 0,1,2,3$.

§3. Theoretical Aspects

In order to better understand univariate thinning, and in particular why AT1 is almost as good as AT0, we study the antipicated error used in AT1.

Notice that for any $i \in \{2,3,\ldots,n-1\}$, and $x \in [y_{i-1}, y_{i+1}] \cap X$, the error in the linear interpolation at y_{i-1}, y_{i+1} is given by

$$L(f,\{y_{i-1},y_{i+1}\})(x) - f(x) = (x - y_{i-1})(y_{i+1} - x) f[y_{i-1}, x, y_{i+1}],$$

where $f[a,b,c]$ denotes the usual second order divided difference of the function f at the abscissae a, b, c. It follows that

$$e_1(y_i) = \max_{x \in [y_{i-1},y_{i+1}] \cap X} (x - y_{i-1})(y_{i+1} - x) \big| f[y_{i-1}, x, y_{i+1}] \big|.$$

For f a quadratic polynomial, this identity leads us to a relationship between adaptive and non-adaptive thinning.

Proposition 3.1. *If f is a quadratic polynomial, then the adaptive univariate thinning algorithms AT1 and AT2 are non-adaptive. A point y_i in Y is removable if and only if*

$$\max_{x \in [y_{i-1},y_{i+1}] \cap X} (x - y_{i-1})(y_{i+1} - x) = \min_{1<j<n} \max_{x \in [y_{j-1},y_{j+1}] \cap X} (x - y_{j-1})(y_{j+1} - x) \tag{8}$$

in AT1, and

$$(y_i - y_{i-1})(y_{i+1} - y_i) = \min_{1<j<n} (y_j - y_{j-1})(y_{j+1} - y_j) \tag{9}$$

in AT2.

Proof: Since for any $a,b,c \in \mathbb{R}$, the divided difference $f[a,b,c]$ is a constant, the anticipated errors (5) and (6) reduce, after a scaling, to

$$\max_{x \in [y_{i-1},y_{i+1}] \cap X} (x - y_{i-1})(y_{i+1} - x),$$

and $(y_i - y_{i-1})(y_{i+1} - y_i)$ respectively. $\square$

Note that AT2 reduces to NAT, in case f is a quadratic polynomial. The criteria (8) and (9) clearly favour well distributed subsets of data: a data point y_i is likely to be removed if it is close to its neighbours. To see this in the case of (8), we replace the discrete set X by the whole interval $[a,b]$, and the dicrete antipicated error $e_1(y_i)$ becomes

$$e_a(y_i) = \max_{x\in[y_{i-1},y_{i+1}]} |L(f,\{y_{i-1},y_{i+1}\})(x) - f(x)|.$$

Thus, if f is a quadratic polynomial, then

$$e_a(y_i) = \frac{1}{8}(y_{i+1} - y_{i-1})^2 |f''|,$$

and y_i is removable if and only if

$$y_{i+1} - y_{i-1} = \min_{1<j<n} (y_{j+1} - y_{j-1}).$$

This is the removal criterion of the non-adaptive Thinning Algorithm 3 of [3].

Next we give an explanation of why minimizing the anticipated error $e_1(\cdot)$ instead of the actual error $e_0(\cdot)$ in the thinning algorithm AT1 results in a good algorithm. We do it by considering convex functions f. First we establish a lemma.

Lemma 3.2. *Suppose f is convex, and let Y be any subset of X of the form (1). Then for any $i \in \{2,\dots,n-1\}$,*

$$e_0(y_i) = \max\{e_1(y_i), E(Y;X;f)\}. \tag{10}$$

Proof: Due to the convexity of f, we have

$$e_1(y_i) = e_{[y_{i-1},y_{i+1}]} \ge \max\{e_{[y_{i-1},y_i]}, e_{[y_i,y_{i+1}]}\},$$

and since

$$e_0(y_i) = E(Y\setminus y_i;X;f) = \max\{e_1(y_i), \max_{\substack{k=1,\dots,n-1\\ k\neq i-1,i}} e_{[y_k,y_{k+1}]}\}, \tag{11}$$

we find

$$e_0(y_i) = \max\{e_1(y_i), \max_{k=1,\dots,n-1} e_{[y_k,y_{k+1}]}\} = \max\{e_1(y_i), E(Y;X;f)\}. \quad \square$$

Proposition 3.3. *Suppose f is convex and let Y be any subset of X of the form (1). Then for any $i,j \in \{2,\dots,n-1\}$,*

$$e_1(y_i) \le e_1(y_j) \qquad \Longrightarrow \qquad e_0(y_i) \le e_0(y_j). \tag{12}$$

Proof: From (10), we have

$$e_0(y_i) = \max\{e_1(y_i), E(Y;X;f)\} \le \max\{e_1(y_j), E(Y;X;f)\} = e_0(y_j). \quad \square$$

Thus for convex data, the thinning algorithm AT1 performs as AT0. We show in the next example that there are arbitrary subsets of non-convex data for which (12) does not hold.

Example 3.4. *((12) does not hold for non-convex data.) Let*

$$X = (x_1, \dots, x_7) = (1, 2, 3, 4, 5, 6, 7),$$

and let the non-convex function f be the piecewise linear interpolant over X satisfying $f(x_i) = f_i$, where

$$(f_1, \dots, f_7) = (0, 0, -1, 1, 0, 0, 0).$$

Consider the subset $Y = X \setminus x_4$. Then $e_1(x_6) = 0$ and $e_1(x_3) = 1$, while $e_0(x_6) = 3/2$ and $e_0(x_3) = 1$.

Note however, that if the thinning algorithm AT1 were applied to this example, Y would not be the subset generated by the first removal as the first point to be removed would be x_6. We have not been able to construct an example of a data set X and a non-convex function f where AT0 acts differently from AT1 at any stage of the thinning algorithm. In the absence of such an example, and since typical data sets are locally convex or concave in large regions, i.e. there are relatively few inflection points, we arrive at the conclusion, supported by our numerical experiments, that AT1 is a good, computationally inexpensive thinning algorithm.

§4. Algorithmic Aspects

In this section, we discuss details concerning our implementation of the four thinning algorithms AT0, AT1, AT2, and NAT. Moreover, we shall compute their asymptotic complexity.

We first discuss AT2 and NAT. The interior points of the current set Y are stored in a heap, according to the sizes of their anticipated errors; $e_2(.)$ for AT2 and $e_3(.)$ for NAT. A heap is a binary tree which can be used for the implementation of a priority queue. Each point y in the heap bears its anticipated error as its significance value. Due to the heap condition, the significance of a node is smaller than the significances of its two children. Therefore, the root of the heap contains a removable point. It is well-known [7] that each insertion, removal, or update of one node in the heap costs $O(\log n)$ operations, where n is the number of nodes in the heap. In consequence, building the initial heap costs $O(N \log N)$ operations.

Now suppose Y is of the form (1), of size n. The number of points already removed is $N - n$. We perform Step 2) of the thinning algorithm of Section 2 as follows.

1) Pop the root y_i from the heap.
2) Compute $e(y_{i-1}; Y \setminus y_i)$ and $e(y_{i+1}; Y \setminus y_i)$ and update the heap.
3) Let $Y = Y \setminus y_i$.

As regards the number of operations, Steps 1) and 2) both require $O(\log n)$ operations, while Step 3) requires $O(1)$ operations. Therefore, summing the costs of Steps 1) to 3) for all n, we find that the total cost of the thinning algorithms AT2 and NAT is $O(N \log N)$.

Next we describe Step 2) of the thinning algorithm in Section 2 for AT1, which is somewhat more complicated than the previous ones. For this algorithm, we store $\{e_1(y) : y \in Y\}$ in a heap where the root contains the removable point for AT1.

1) Pop the root y_i from the heap.
2) Attach y_i and the previously removed points attached to the intervals $[y_{i-1}, y_i]$ and $[y_i, y_{i+1}]$ to the new interval $[y_{i-1}, y_{i+1}]$ (generated after the removal of y_i).
3) Compute $e_1(y_{i-1}; Y \setminus y_i)$ and $e_1(y_{i+1}; Y \setminus y_i)$ and update the e_1-heap.
4) Let $Y = Y \setminus y_i$.

Thus, during the adaptive thinning algorithm AT1, each of the points already removed is attached to an interval corresponding to the current subset. These attachments facilitate the computation of the anticipated error of the neighbouring points of y_i in Y, whose anticipated errors in $Y \setminus y_i$ differ from their anticipated errors in Y.

As regards the number of operations, Steps 1) and 4) are as in the previous algorithm. Step 2) requires $O((N-n)/n)$ operations under the additional assumption that the number of points attached to an interval is of the order of $(N-n)/n$. The computation of the anticipated error in step (3) is also $O((N-n)/n)$. So altogether the total cost is $O(N \log N)$, just as for AT2 and NAT, though with a higher constant.

The algorithm for AT0 is a variant of the algorithm for AT1, but is more complicated. Yet it can be organized so that the total cost remains $O(N \log N)$. We now employ two heaps, the first of which is the e_1-heap we used for AT1. The second heap, which we call the I-heap, consists of the values $\{e_{[y_i, y_{i+1}]}; i = 1, 2, \ldots, n-1\}$ so that the root of the heap points to the *maximal* element. Using the identity (11), it can easily be shown that there is always a removable point amongst the three points y_i, the root of the e_1-heap, and y_j and y_{j+1}, where $[y_j, y_{j+1}]$ is the root of the I-heap. Thus, it is only necessary to compute $e_0(.)$ at these three points and take the minimum, and using the two heaps, this can be achieved in just $O(\log n)$ operations. The update of the e_1-heap after the removal requires $O(\log n)$ operations as in the algorithm for AT1, and the update of the I-heap also requires just $O(\log n)$ operations. Thus the thinning algorithm AT0 requires $O(N \log N)$ operations, but with a larger constant than for AT1.

§5. Numerical Examples

We have implemented the four thinning algorithms AT0, AT1, AT2, and NAT corresponding to the error measures (4), (5), (6), (7) in Section 2. In this section we compare the performance of these algorithms in terms of their approximation error and computational costs. For the purpose of illustration,

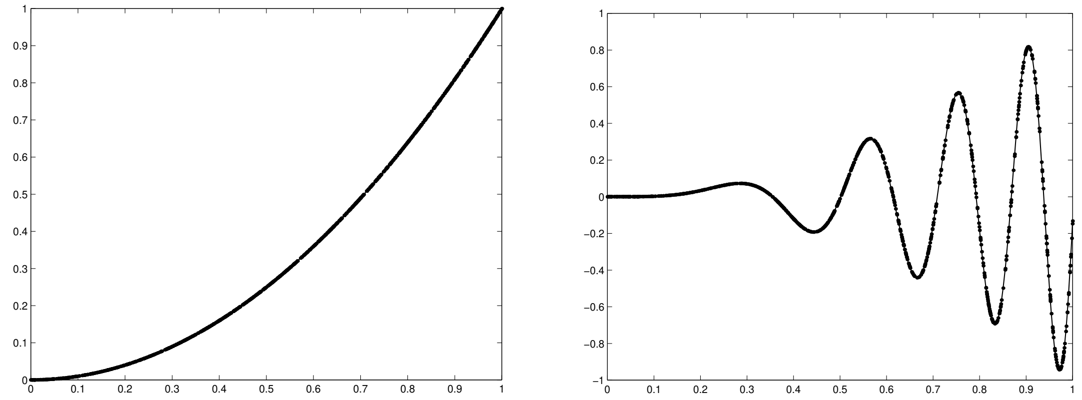

Fig. 1. Data sets sampled from f_1 (left) and f_2 (right).

we use the two test functions $f_1(x) = x^2$ and $f_2(x) = x^2 \sin(25x^2)$, which we sampled at a set X of 1000 randomly chosen points in the unit interval $(0, 1)$, together with the two boundary points $0, 1$, cf. Figure 1.

We have computed all subsets $X_n \subset X$ for $n = N, N-1, \dots, 2$, where $N = 1002$, output by the four thinning algorithms. For each test case, we have recorded both the resulting max error $E_\infty(X_n) \equiv E(X_n; X; f)$ and ℓ_2 error

$$E_2^2(X_n) \equiv E_2^2(X_n; X; f) = \sum_{x \in X} |L(f, X_n)(x) - f(x)|^2.$$

For the test cases involving the quadratic function f_1, we observe that the four subsets X_n obtained by the four thinning algorithms have nearly equal approximation errors, $E_\infty(X_n)$ and $E_2(X_n)$. For $n = 22$, the resulting values are displayed in Table 1 which also shows the required computational costs in CPU time. Not surprisingly, the thinning method NAT is the fastest, followed by AT2 and AT1, whereas AT0 is the slowest. Table 1 also shows the *mesh ratio*,

$$\rho(\{y_1, \cdots, y_n\}) = \min_{0<j<n} |y_{j+1} - y_j| / \max_{0<j<n} |y_{j+1} - y_j|,$$

for each subset. From the values $\rho(X_{22})$ for the four subsets, we conclude that these subsets are well distributed in $[0, 1]$. Figure 2 shows the subset X_{22} selected by the methods AT0, AT1 (left) and AT2, NAT (right).

Method	$E_\infty(X_{22})$	$E_2(X_{22})$	$\rho(X_{22})$	CPU
AT0	0.0013880	0.0166962	0.4029	0.70
AT1	0.0013880	0.0166962	0.4029	0.49
AT2	0.0010135	0.0164082	0.4081	0.27
NAT	0.0010135	0.0164082	0.4081	0.26

Tab. 1. Thinning to 22 points with f_1.

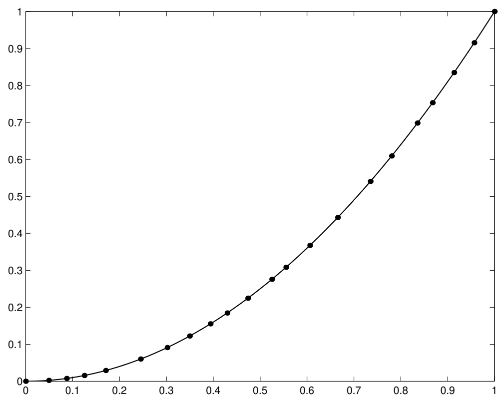

Fig. 2. The subsets X_{22} output by AT0, AT1 (left) and AT2, NAT (right).

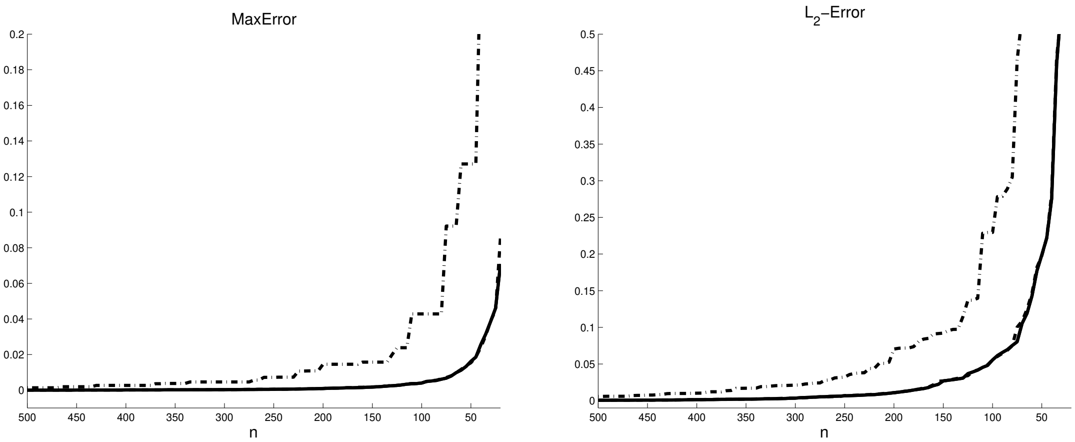

Fig. 3. Error of AT0,AT1 (solid), AT2 (dashed) and NAT (dash-dotted).

As expected from the theoretical results of Section 3, we see in Table 1 that AT0 is identical with AT1 since f_1 is convex. Also, AT2 and NAT are identical since f_1 is a quadratic polynomial. Thus all four algorithms are non-adaptive, and generate well distributed subsets.

Now let us turn to the test case involving the oscillating function f_2. In contrast to the results for f_1, we find that the three adaptive thinning methods AT0, AT1, and AT2 are, especially for the selection of small subsets, clearly superior to NAT in terms of approximation error. This is confirmed by Figure 3 showing the four graphs of $E_\infty(X_n; X; f_2)$ and $E_2(X_n; X; f_2)$, for $n = 500, 499, \ldots, 22$.

Observe from Figure 3 that the approximation behaviour of the three methods AT0, AT1, and AT2 is quite similar. In fact, we found that for any n the two subsets X_n output by AT0 and AT1 coincide. Taking a closer look at the approximation errors of AT1 and AT2, we see from Figure 4 that for very large numbers of removed points, AT1 is superior to AT2, in terms of the error $E_\infty(X_n; X; f_2)$. The trade-off is that AT1 typically required about 60% more CPU time than AT2.

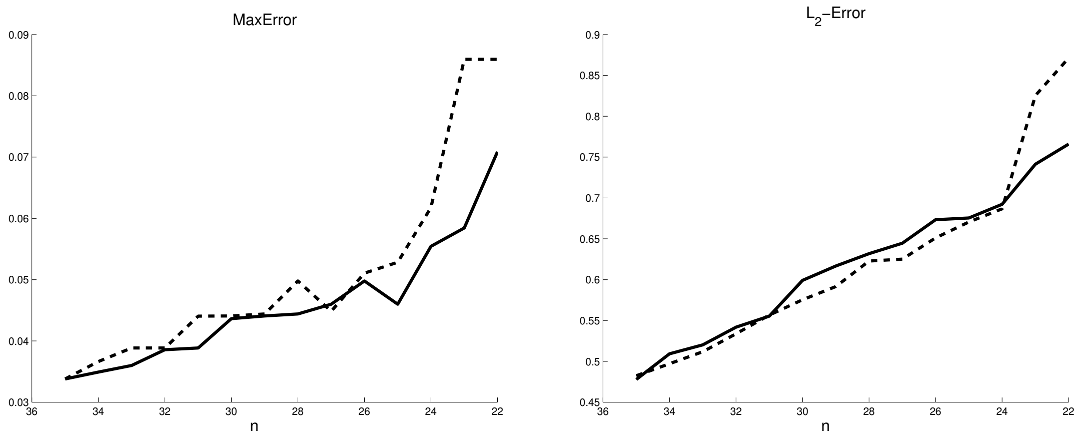

Fig. 4. Error of AT1 (solid), and AT2 (dashed) for $n = 35, 34, \ldots, 22$.

Finally, we wish to demonstrate the utility of adaptive thinning for a class of approximation methods other than piecewise linear interpolation. For subsets Y generated by the thinning algorithms, we computed the least squares approximation $s^*_{\phi,Y} \in S_{\phi,Y}$ satisfying

$$\eta^2_{\phi,2}(Y) = \sum_{x \in X} |s^*_{\phi,Y}(x) - f(x)|^2 = \min_{s \in S_{\phi,Y}} \sum_{x \in X} |s(x) - f(x)|^2,$$

where $S_{\phi,Y} = \text{span}\,\{\phi(|\cdot - y|) : y \in Y\}$ denotes the linear space of all linear combinations of Y-translates of the multiquadrics $\phi(r) = \sqrt{c^2 + r^2}$, $c > 0$ (see [5] for more details). This gives two additional criteria $\eta_{\phi,2}(Y)$ and

$$\eta_{\phi,\infty}(Y) = \max_{x \in X} |s^*_{\phi,Y}(x) - f(x)|,$$

for judging the *quality* of a subset Y of X. In order to show one concrete example, we let $c = 0.2$, $n = 22$. Table 2 reflects the numerical results, and Figure 5 show the two subsets $Y = X_{22}$ selected by the method AT1 (left) and NAT (right) along with the graphs of their corresponding least squares approximations $s^*_{\phi,Y}$.

Method	$E_\infty(X_{22})$	$E_2(X_{22})$	$\eta_{\phi,\infty}(X_{22})$	$\eta_{\phi,2}(X_{22})$
AT1	0.07088	0.76592	0.0196591	0.091160
NAT	0.84983	5.76083	0.0497937	0.567353

Tab. 2. Thinning to 22 points with f_2.

Table 2 indicates that small subsets of X output by an adaptive thinning algorithm can serve as good sets of centres for approximating the data $(x, f(x)) : x \in X$, by a sum of translates of ϕ to the chosen centres, and that these approximations are superior to the piecewise linear interpolants on these subsets.

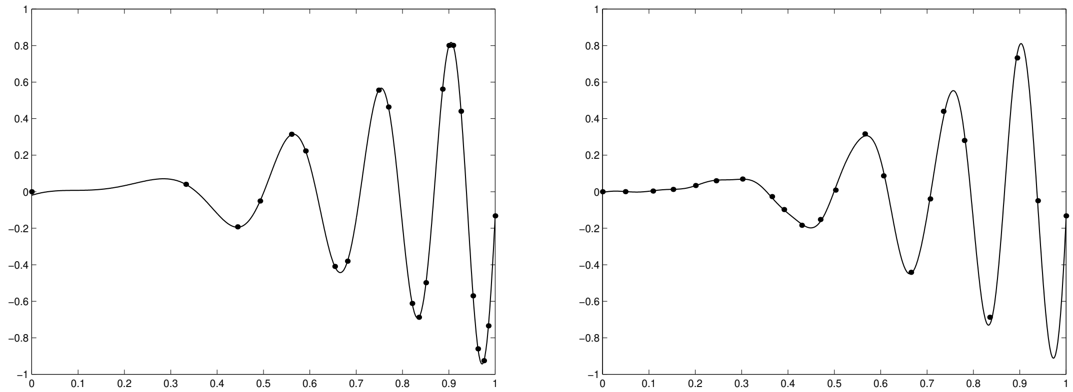

Fig. 5. Least squares approximation, AT1 (left), NAT (right).

Acknowledgments. The authors were partly supported by the European Union within the project MINGLE (Multiresolution in Geometric Modelling), contract no. HPRN-CT-1999-00117.

References

1. Dyn, N., M. S. Floater, and A. Iske, Adaptive thinning of bivariate scattered data, preprint.
2. Floater, M. S. and A. Iske, Multistep Scattered Data Interpolation using Compactly Supported Radial Basis Functions, J. Comp. Appl. Math. **73** (1996), 65–78.
3. Floater, M. S. and A. Iske, Thinning algorithms for scattered data interpolation, BIT **38** (1998), 705–720.
4. Heckbert, P. S. and M. Garland, Survey of Surface Simplification Algorithms, Technical Report, Computer Science Dept., Carnegie Mellon University, 1997.
5. Iske, A., Reconstruction of smooth signals from irregular samples by using radial basis function approximation, in *Proceedings of the 1999 International Workshop on Sampling Theory and Applications*, Y. Lyubarskii (ed.), The Norwegian University of Science and Technology, Trondheim, 1999, 82–87.
6. Lyche, T., Knot removal for spline curves and surfaces, in *Approximation Theory VII*, E. W. Cheney, C. K. Chui, and L. L. Schumaker, (eds.), Academic Press, Boston, 1993, 207–227.
7. R. Sedgewick, *Algorithms*, Addison-Wesley, Reading, MA, 1983.

Nira Dyn
School of Mathematical Sciences
Sackler Faculty of Exact Sciences
Tel-Aviv University
Tel Aviv 69978, Israel
`niradyn@math.tau.ac.il`

Michael S. Floater
SINTEF
Post Box 124, Blindern
N-0314 Oslo, Norway
mif@math.sintef.no

Armin Iske
Technische Universität München
Zentrum Mathematik
D-80290 München, Germany
iske@ma.tum.de