An Optimal Linear Time Algorithm for Quasi-Monotonic Segmentation

Daniel Lemire
University of Quebec at Montreal (UQÀM)
4750, avenue Henri-Julien
Montréal, Qc, Canada, H2T 3E4
lemire@ondelette.com

Martin Brooks, Yuhong Yan
National Research Council of Canada
1200 Montreal Road
Ottawa, ON, Canada, K1A 0R6
First.Last@nrc.gc.ca

Abstract

Monotonicity is a simple yet significant qualitative characteristic. We consider the problem of segmenting an array in up to K segments. We want segments to be as monotonic as possible and to alternate signs. We propose a quality metric for this problem, present an optimal linear time algorithm based on novel formalism, and compare experimentally its performance to a linear time top-down regression algorithm. We show that our algorithm is faster and more accurate. Applications include pattern recognition and qualitative modeling.

1 Introduction

Monotonicity is one of the most natural and important qualitative properties for sequences of data points. It is easy to determine where the values are strictly going up or down, but we only want to identify significant monotonicity. For example, the drop from 2 to 1.9 in the array $0,1,2,1.9,3,4$ might not be significant and might even be noise-related. The quasi-monotonic segmentation problem is to determine where the data is approximatively increasing or decreasing.

Some segmentation algorithms give a segmentation having no more than K segments while attempting to minimize the error ε; other algorithms attempt to minimize the number of segments (K) given an upper bound on the error ε. We are concerned with the first type of algorithm in this paper.

Using dynamic programming or other approaches, most segmentation problems can be solved in time $O(n^2)$. Other solutions to this problem using machine learning to classify the pairs of data points [7] are even less favorable since they have higher complexity. However, it is common for sequence of data points to be massive and segmentation algorithms have to have complexity close to $O(n)$ to be competitive. While approximate linear regression segmentation algorithms can be $O(n)$, we show that using a linear regression error to segment according to monotonicity is not an ideal solution.

We present a metric for the quasi-monotonic segmentation problem called the Optimal Monotonic Approximation Function Error (OMAFE); this metric differs from previously introduced OPMAFE metric [3] since it applies to all segmentations and not just "extremal" segmentations. We formalize the novel concept of a maximal $*$-pair and shows that it can be used to define a unique labelling of the extrema leading to a novel optimal segmentation algorithm. We also present an optimal linear time algorithm to solve the quasi-monotonic segmentation problem together with an experimental comparison to quantify the benefits of our algorithm.

2 Monotonicity Error Metric (OMAFE)

Suppose n samples noted $F : D = \{x_1, \ldots, x_n\} \rightarrow \mathbb{R}$ with $x_1 < x_2 < \ldots x_n$. We define, $F_{|[a,b]}$ as the restriction of F over $D \cap [a,b]$. We seek the best monotonic (increasing or decreasing) function $f : \mathbb{R} \rightarrow \mathbb{R}$ approximating F. Let $\Omega_{\uparrow}$ (resp. $\Omega_{\downarrow}$) be the set of all monotonic increasing (resp.

decreasing) functions. The **Optimal Monotonic Approximation Function Error (OMAFE)** is $\min_{f\in\Omega}\max_{x\in D}|f-F|$ where Ω is either $\Omega_\uparrow$ or $\Omega_\downarrow$.

The segmentation of a set D is a sequence $S = X_1,\ldots,X_K$ of intervals in $\mathbb{R}$ with $[\min D,\max D] = \bigcup_i X_i$ such that $\max X_i = \min X_{i+1} \in D$ and $X_i \cap X_j = \emptyset$ for $j \neq i+1, i, i-1$. Alternatively, we can define a segmentation from the set of points $X_i \cap X_{i+1} = \{y_{i+1}\}$, $y_1 = \min X_1$, and $y_{K+1} = \max X_K$. Given $F : \{x_1,\ldots,x_n\} \to \mathbb{R}$ and a segmentation, the Optimal Monotonic Approximation Function Error (OMAFE) of the segmentation is $\max_i \text{OMAFE}(F_{|X_i})$ where the monotonicity type (increasing or decreasing) of the segment X_i is determined by the sign of $F(\max X_i) - F(\min X_i)$. Whenever $F(\max X_i) = F(\min X_i)$, we say the segment has no direction and the best monotonic approximation is just the flat function having value $(\max F_{|X_i} - \min F_{|X_i})/2$. The error is computed over each interval independently and so, optimal monotonic approximation functions are not required to agree at $\max X_i = \min X_{i+1}$. Segmentations should alternate between increasing and decreasing, otherwise sequences such as $0,2,1,0,2$ can be segmented as two increasing segments $0,2,1$ and $1,0,2$: we consider it is natural to aggregate segments with the same monotonicity.

We solve for the best monotonic function as follows. If we seek the best monotonic increasing function, we first define $\overline{f}_\uparrow(x) = \max\{F(y) : y \le x\}$ (the maximum of all previous values) and $\underline{f}_\uparrow(x) = \min\{F(y) : y \ge x\}$ (the minimum of all values to come). If we seek the best monotonic decreasing function, we define $\overline{f}_\downarrow(x) = \max\{F(y) : y \ge x\}$ (the maximum of all values to come) and $\underline{f}_\downarrow(x) = \min\{F(y) : y \le x\}$ (the minimum of all previous values). These functions, which can be computed in linear time, are all we need to solve for the best approximation function as shown by the next theorem which is a well-known result [2, 6].

Theorem 1 *Given $F : D = \{x_1,\ldots,x_n\} \to \mathbb{R}$, a best monotonic increasing approximation function to F is $f_\uparrow = (\overline{f}_\uparrow + \underline{f}_\uparrow)/2$ and a best monotonic decreasing approximation function is $f_\downarrow = (\overline{f}_\downarrow + \underline{f}_\downarrow)/2$. The corresponding error (OMAFE) is $\max_{x\in D}(|\overline{f}_\uparrow(x) - \underline{f}_\uparrow(x)|)/2$ or $\max_{x\in D}(|\overline{f}_\downarrow(x) - \underline{f}_\downarrow(x)|)/2$ respectively.*

The implementation of the algorithm suggested by the theorem is straight-forward. Given a segmentation, we can compute the OMAFE in $O(n)$ time using at most two passes.

3 A Scale-Based Algorithm for Quasi-Monotonic Segmentation

We use the following proposition to prove that the segmentations we generate are optimal (see Theorem 2 on page 4).

Proposition 1 *A segmentation $y_1,\ldots,y_{K+1}$ of $F : D = \{x_1,\ldots,x_n\} \to \mathbb{R}$ with alternating monotonicity has a minimal OMAFE ε for a number of alternating segments K if*

A. $F(y_i) = \max F([y_{i-1},y_{i+1}])$ or $F(y_i) = \min F([y_{i-1},y_{i+1}])$ for $i = 2,\ldots,K$;

B. in all intervals $[y_i,y_{i+1}]$ for $i = 1,\ldots,K$, there exists z_1,z_2 such that $|F(z_2) - F(z_1)| > 2\varepsilon$.

Proof. Let the original segmentation be the intervals $S_1,\ldots,S_K$ and consider a new segmentation with intervals $T_1,\ldots,T_K$. Assume that the new segmentation has lower error (as given by OMAFE). Let $S_i = [y_i,y_{i+1}]$ and $T_i = [y'_i,y'_{i+1}]$.

If any segment T_m contains a segment S_j, then the existence of z_1,z_2 in $[y_j,y_{j+1}]$ such that $|F(z_2) - F(z_1)| > 2\varepsilon$ and $\text{OMAFE}(T_m) \le \varepsilon$ implies that T_m and S_j have the same monotonicity.

We show that each pair of intervals S_i, T_i has nonempty intersection. Suppose not, and let i be the smallest index such that $S_i \subset T_{i-1}$. Since S_i and T_{i-1} have the same monotonicity, for each $j < i$, S_j and T_j have opposite monotonicity. Now consider the $i-1$ intervals $T_1,\ldots,T_{i-1}$ and the i points $y_1,\ldots,y_i$. At least one interval contains two consecutive points; choose the largest $j < i$ such that T_j contains y_j,y_{j+1}. But then $S_j \subset T_j$, contradicting at least one of the assumptions $|F(z_2) - F(z_1)| > 2\varepsilon$ for $z_1,z_2 \in S_i$ and $\text{OMAFE}(T_j) \le \varepsilon$.

It now follows that each pair of intervals S_i,T_i has the same monotonicity.

Since $\text{OMAFE}(T) < \text{OMAFE}(S)$, we can choose an index j such that $\text{OMAFE}(T_j) < \text{OMAFE}(S_j)$. We show that there exists another index p such that $\text{OMAFE}(T_p) \ge \text{OMAFE}(S_j)$, thus contradicting $\text{OMAFE}(T) < \text{OMAFE}(S)$. Suppose S_j is increasing; the proof is similar for the opposite case. Then there exist $x < z \in S_j$ such that $F(x) - F(z) = 2 \times OMAFE(S_j)$. From $\text{OMAFE}(T_j) < \text{OMAFE}(S_j)$ it follows that at least

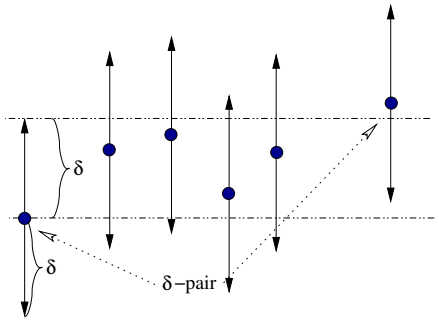

Figure 1. A δ-pair.

one of x or z lies in $S_j - T_j$, and hence $F(x) - F(y_j) \geq 2 \times \text{OMAFE}(S_j)$ or $F(y_{j+1}) - F(z) \geq 2 \times \text{OMAFE}(S_j)$. Thus $\text{OMAFE}(T_p) \geq \text{OMAFE}(S_j)$ for either $p = j-1$ or $p = j+1$. □

For simplicity, we assume F has no consecutive equal values, i.e. $F(x_i) \neq F(x_{i+1})$ for $i = 1, \ldots, n-1$; our algorithms assume all but one of consecutive equal values values have been removed. We say x_i is a maximum if $i \neq 1$ implies $F(x_i) > F(x_{i-1})$ and if $i \neq n$ implies $F(x_i) > F(x_{i+1})$. Minima are defined similarly.

Our mathematical approach is based on the concept of δ-pair [2] (see Fig. 1):

Definition 1 *The tuple x, y ($x < y \in D$) is a δ-pair (or a pair of scale δ) for F if $|F(y) - F(x)| \geq \delta$ and for all $z \in D$, $x < z < y$ implies $|F(z) - F(x)| < \delta$ and $|F(y) - F(z)| < \delta$. A δ-pair's direction is increasing or decreasing according to whether $F(y) > F(x)$ or $F(y) < F(x)$.*

δ-Pairs having opposite directions cannot overlap but they may share an end point. δ-Pairs of the same direction may overlap, but may not be nested. We use the term "$*$-pair" to indicate a δ-pair having an unspecified δ. We say that a $*$-pair is significant at scale δ if it is of scale δ' for $\delta' \geq \delta$.

We define δ-monotonicity as follows:

Definition 2 *Let X be an interval, F is δ-monotonic on X if all δ-pairs in X have the same direction; F is strictly δ-monotonic when there exists at least one such δ-pair. In this case:*

- *F is δ-increasing on X if X contains an increasing δ-pair.*
- *F is δ-decreasing on X if X contains a decreasing δ-pair.*

A δ-monotonic interval X satisfies $\text{OMAFE}(X) < \delta/2$.

We say that a $*$-pair x, y is **maximal** if whenever z_1, z_2 is a $*$-pair of a larger scale in the same direction containing x, y, then there exists a $*$-pair w_1, w_2 of an opposite direction contained in z_1, z_2 and containing x, y. For example, the sequence $1, 3, 2, 4$ has 2 maximal $*$-pairs: $1, 4$ and $3, 2$. Maximal $*$-pairs of opposite direction may share a common point, whereas maximal $*$-pairs of the same direction may not. Maximal $*$-pairs cannot overlap, meaning that it cannot be the case that exactly one end point of a maximal $*$-pair lies strictly between the end points of another maximal $*$-pair; either neither point lies strictly between or both do. In the case that both do, we say that the one maximal $*$-pair properly contains the other. All $*$-pairs must be contained in a maximal $*$-pair.

Lemma 1 *The smallest maximal $*$-pair containing a $*$-pair must be of the same direction.*

Proof. Suppose a $*$-pair is immediately contained in a maximal $*$-pair W. Suppose W is not in the same direction, then within W, seek the largest $*$-pair in the same direction as P and containing P, then it must be a maximal $*$-pair in D since maximal $*$-pairs of different directions cannot overlap. □

The first and second point of a maximal $*$-pair are extrema and the reverse is true as well as shown by the next lemma.

Lemma 2 *Every extremum is either the first or second point of a maximal $*$-pair.*

Proof. The case $x = x_1$ or $x = x_n$ follows by inspection. Otherwise, x is the end point of a left and a right $*$-pair. Each $*$-pair must immediately belong to a maximal $*$-pair of same direction: a $*$-pair P is contained in a maximal $*$-pair M of same direction and there is no maximal $*$-pair M' of opposite direction such that $P \subset M' \subset M$. Let M^l and M^r be the maximal $*$-pairs immediately containing the left and right $*$-pair of x. Suppose neither M^l and M^r have x as a end point. Suppose $M^l \subset M^r$, then the right $*$-pair is not immediately contained in M^r, a contradiction. The result follows by symmetry. □

Our approach is to label each extremum in F with a scale parameter δ saying that this extremum is "significant" at scale δ and below. Our intuition is that by picking extrema at scale δ, we should have a segmentation having error less than $\delta/2$.

Definition 3 *The scale labelling of an extremum x is the maximum of the scales of the maximal $*$-pairs for which it is an end point.*

For example, given the sequence $1,3,2,4$ with 2 maximal $*$-pairs ($1,4$ and $3,2$), we would give the following labels in order $4,1,1,4$.

Definition 4 *Given $\delta > 0$, a maximal alternating sequence of δ-extrema $Y = y_1 \ldots y_{K+1}$ is a sequence of extrema each having scale label at least δ, having alternating types (maximum/minimum), and such that there exists no sequence properly containing Y having these same properties. From Y we define a maximal alternating δ-segmentation of D by segmenting at the points $x_1, y_2 \ldots y_K, x_n$.*

Theorem 2 *Given $\delta > 0$, let $P = S_1 \ldots S_K$ be a maximal alternating δ-segmentation derived from maximal alternating sequence $y_1 \ldots y_{K+1}$ of δ-extrema. Then any alternating segmentation Q having OMAFE(Q) $<$ OMAFE(P) has at least $K+1$ segments.*

Proof. We show that conditions A and B of Proposition 1 are satisfied with $\varepsilon = \text{OMAFE}(P)$.

First we show that each segment S_i is δ-monotone; from this we conclude that $\text{OMAFE}(P) < \delta/2$. Intervals $[x_1, y_1]$ and $[y_K, x_n]$ contain no maximal $*$-pairs of scale δ or larger, and therefore contain no $*$-pairs of scale δ or larger. Similarly, no $[y_i, y_{i+1}]$ contains an opposite-direction significant $*$-pair.

Condition A: Follows from δ-monotonicity of each S_i and maximal $*$-pairs not overlapping.

Condition B: We show that $|F(y_{i+1}) - F(y_i)| \geq \delta > 2 \times \text{OMAFE}(P)$. If $i = 1$, then y_i must begin an maximal $*$-pair, and the maximal $*$-pair must end with y_{i+1} since maximal $*$-pairs cannot overlap. The case $i+1 = k$ is similar. Otherwise, since maximal $*$-pairs cannot overlap, each y_i, y_{i+1} is either a maximal $*$-pair of scale δ or larger or there exist indices j and k, $j < i$ and $k > i+1$ such that y_j, y_i is a maximal $*$-pair of scale at least δ, and y_{i+1}, y_k is a maximal $*$-pair of scale at least δ. These two maximal $*$-pairs have the same direction, and that this is opposite to the direction of $[y_i y_{i+1}]$. Now suppose $|F(y_i) - F(y_{i+1})| < \delta$. Then y_j, y_k is a $*$-pair properly containing y_j, y_i and y_{i+1}, y_k. But neither y_j, y_i nor y_{i+1}, y_k can be properly contained in a $*$-pair of opposite direction lying within y_j, y_k, thus contradicting their maximality and proving the claim. □

Algorithmic simplicity and efficiency is achieved when each maximal alternating sequence of δ-extrema is simply the sequence of all extrema labelled by at least δ. This is generally not the case, for example the sequence $0,10,9,10,0$ is scale labelled $10,10,1,10,10$. However, a simple relabelling of certain extrema can achieve the desired effect. Consider two same-sense extrema $z_1 < z_2$ such that lying between them there exists no extremum having scale at least as large as the minimum of the two extrema's scales. This is the only situation which causes choice when constructing a maximal alternating sequence of δ-extrema. To eliminate this choice, replace the scale label on z_1 with the largest scale of the opposite-sense extrema lying between them. It is easily seen that in the case just described we must have $F(z_1) = F(z_2)$, since otherwise the point upon which F has the lesser value could not be the endpoint of a maximal $*$-pair. In the next section, Algorithm 1 incorporates this re-labelling making Algorithm 2 simple and efficient.

3.1 Computing a Scale Labelling Efficiently

Algorithm 1 produces a scale labelling in linear time. Extrema from the original data are visited in order, and they alternate (maxima/minima) since we only pick one of the values when there are repeated values (such as $1,1,1$).

The algorithm has a main loop (lines 5 to 12) where it labels extrema as it identifies extremal $*$-pairs, and stack the extrema it cannot immediately label. At all times, the stack (line 3) contains minima and maxima in **strictly** increasing and decreasing order respectively. Also at all times, the last two extrema at the bottom of the stack are the absolute maximum and absolute minimum (found so far). Observe that we can only label an extrema as we find new extremal $*$-pairs (lines 7, 10, and 14).

- If the stack is empty or contains only one extremum, we simply add the new extremum (line 12).
- If there are only 2 extrema z_1, z_2 in the stack and we found either a new absolute maximum or new absolute minimum (z_3), we can pop and label the oldest one (z_1) (lines 9, 10, and 11) because the old pair (z_1, z_2) forms a maximal $*$-pair and thus must be bounded

by extrema having at least the same scale: but the oldest value (z_1) doesn't belong to a larger maximal $*$-pair. This applies even when $z_3 = z_1$. Otherwise, if there are only 2 extrema z_1, z_2 in the stack and the new extrema z_3 satisfies $z_3 \in (\min(z_1,z_2), \max(z_1,z_2))$, then we add it to the stack: it is not yet possible to label any of these values.

- While the stack contains more than 2 extrema (lines 6, 7 and 8), we consider the last three points on the stack (s_3, s_2, s_1) where s_1 is the last point added. Let z be the value of the new extrema. If $z \in (\min(s_1,s_2), \max(s_1,s_2))$, then it is simply added to the stack since we cannot yet label any of these points; we exit the while loop. Otherwise, we have a new maximum exceeding or matching the previous one on stack (or a new minimum lower or equal to the previous one on stack), and hence s_1, s_2 is a maximal $*$-pair. If $z \neq s_2$, then s_3, z is a maximal $*$-pair and thus, s_2 cannot be the end of a maximal $*$-pair and s_1 cannot be the beginning of one, hence both s_2 and s_1 are labelled. If $z = s_2$ then we have successive maxima or minima and the same labelling as $z \neq s_2$ applies.

During the "unstacking" (lines 13 and following), we visit a sequence of minima and maxima in increasing and decreasing order respectively and all consecutive values form a maximal $*$-pair so the analysis is easy: we label each point according to the maximal $*$-pair when there is one.

Algorithm 2 shows how once the labelling is complete, one can compute the segmentation in time $O(nK)$ which we consider to be linear time since K is small compared to n. It also uses a fixed amount of memory ($O(K)$). If the segmentation points need to be sorted, then the complexity is $O(nK + K\log K)$.

4 Experimental Results and Comparison to Top-Down Linear Spline

While we have shown that we can compute in linear time an optimal segmentation, it remains to determine whether the relative gains we achieve are significant when compared with a sensible alternative.

As such, we chose the top-down linear spline algorithm [5] which runs in $O(nK^2)$ time. It successively segments the data starting with only one segment, each time picking the segment with the worse linear regression error and finding the best segmentation point; the linear regression is not continuous from one segment to the other. The regression error can be computed in constant time if one has precomputed the range moments [4]. We run through the segments and aggregate consecutive segments having the same sign where the sign of a segment $[y_k, y_{k+1}]$ is defined by the sign of $F(y_{k+1}) - F(y_k)$, setting 0 to be a positive sign (increasing monotonicity).

Algorithm 1 Algorithm to compute the scale labelling in $O(n)$ time. When there are repeated values in the array d, only one of these values is flagged as an extremum.

1: **INPUT:** an array d containing the y values indexed from 1 to n, repeated consecutive values have been removed
2: **OUTPUT:** a scale labelling for all extrema
3: $S \leftarrow$ empty stack, First(S) is the value on top, Second(S) is the second value
4: **define** $\delta(d,S) = |d_{\text{First}(S)} - d_{\text{Second}(S)}|$
5: **for** e index of an extremum in d, e's are visited in increasing order **do**
6: **while** length(S) > 2 and (e is a minimum such that $d_e \leq$ Second(S) or e is a maximum such that $d_e \geq$ Second(S)) **do**
7: label First(S) and Second(S) with $\delta(d,S)$
8: pop stack S twice
9: **if** length(S) is 2 and (e is a minimum such that $d_e \leq$ Second(S) or e is a maximum such that $d_e \geq$ Second(S)) **then**
10: label Second(S) with $\delta(d,S)$
11: remove Second(S) from stack S
12: stack e to S
13: **while** length of $S > 2$ **do**
14: label First(S) with $\delta(d,S)$
15: pop stack S
16: label First(S) and Second(S) with $\delta(d,S)$

4.1 Data Source

ECGs have a well known monotonicity structure with 5 commonly identifiable extrema per pulse (reference points P, Q, R, S, and T) (see Fig. 2) though not all points can be easily identified on all pulses and the exact morphology can vary. As a source of experimental data, we used samples from

Algorithm 2 Given the scale labelling, this algorithm will return a segmentation using at most K segments. It is assumed that there are at least $K+1$ extrema to begin with.

INPUT: an array d containing the y values indexed from 0 to $n-1$
INPUT: K a bound on the number of segments desired
OUTPUT: unsorted segmentation points (a δ-segmentation)
$L \leftarrow$ empty array (capacity $K+3$)
for e is index of an extremum in d having scale δ, e are visited in increasing order **do**
  insert (e,δ) in L so that L is sorted by scale in decreasing order (sort on δ) using binary search
  if length of L is $K+3$ **then**
    pop last(L)
remove all elements of L having the scale of last(L)
RETURN: the indexes in L replacing first one by 1 and last one by n

the MIT-BIH Arrhythmia Database [1]. We only present our results over one sample since we found that results did not vary much between data samples. These ECG recordings used a sampling rate of 360 samples per second per channel with 11-bit resolution (see Fig. 3). We keep 4000 samples (11 seconds) and about 14 pulses, and we do no preprocessing such as baseline correction. We can estimate that a typical pulse has about 5 "easily" identifiable monotonic segments. Hence, out of 14 pulses, we can estimate that there are about 70 significant monotonic segments, some of which match the domain-specific markers (reference points P, Q, R, S, and T). A qualitative description of such data is useful for pattern matching applications, for example.

4.2 Computational Time Versus Segment Budget

We implemented both the scale-based segmentation algorithm and the L_2 norm top-down linear spline approximation algorithm in Python[1] (version 2.3). The implemented scale-based segmentation algorithm is $O(nK)$. The top-down algorithm

[1] Source code and data samples are available from the authors freely.

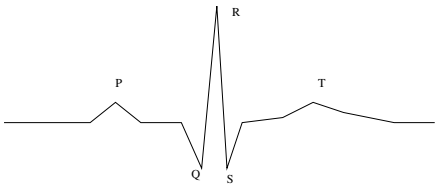

Figure 2. Schema of an ECG pulse with commonly identified reference points (PQRST).

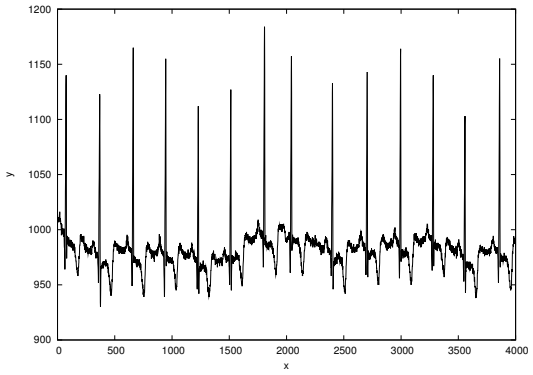

Figure 3. ECG recordings from the MIT-BIH Arrhythmia Database.

used has complexity $O(nK^2)$. Each run was repeated 3 times and the average time is presented in Fig. 4. We chose a single representative sample: results do not vary significantly. The scale-based segmentation implementation is faster than the top-down linear spline approximation implementation and the gains become considerable as K increases.

4.3 OMAFE Versus Segment Budget

We want to determine how well the top-down linear spline algorithm does comparatively. OMAFE is an absolute and not relative error measure, but because the range of the ECGs under consideration is roughly between 950 and 1150, we expect the OMAFE to never exceed 100 by much. The OMAFE with respect to the maximal number of segments (K) is given in Fig. 5: it is a "monotonicity spectrum." By counting on about 5 monotonic segments per pulse with a total of 14 pulses, there should about 70 monotonic segments in the

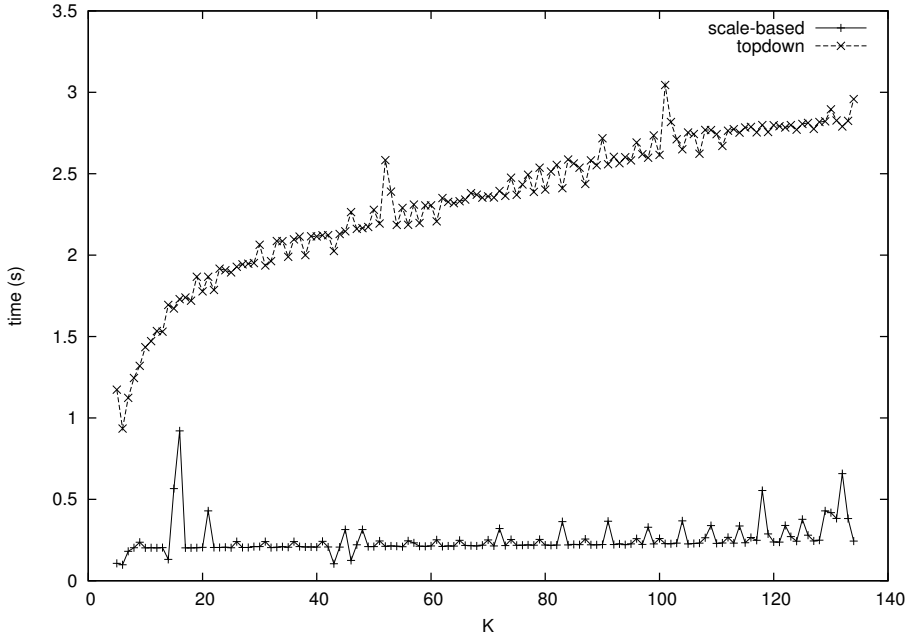

Figure 4. Time versus number of segments K for ECG data.

4000 samples under consideration. We see that the decrease in OMAFE with the addition of new segments starts to level off between 50 and 70 segments as predicted. The addition of new segments past 70 ($K > 70$) has little impact. The scale-based algorithm is optimal, but also at least 3 times more accurate than the top-down algorithm for larger K and this is consistent over other data sets. In fact, the OMAFE becomes practically zero for $K > 80$ whereas the OMAFE of the top-down linear regression algorithm remains at around 20, which is still significant. OMAFE of the scale-based algorithm is a non increasing function of K, a consequence of optimality.

5 Future work

Future work will focus on choosing the optimal number of segments for given applications. We also plan to investigate the applications of the monotonicity spectrum as a robust analysis.

References

[1] A. L. Goldberger *et al.* PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, 101(23):215–220, 2000.

[2] M. Brooks. Approximation complexity for piecewise monotone functions and real data. *Computers and Mathematics with Applications*, 27(8), 1994.

[3] M. Brooks, Y. Yan, and D. Lemire. Scale-based monotonicity analysis in qualitative modelling with flat segments. In *IJCAI'05*, 2005.

[4] D. Lemire. Wavelet-based relative prefix sum methods for range sum queries in data cubes. In *CASCON*. IBM, October 2002.

[5] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, and W. Truppel. Online amnesic algorithm of streaming time series. In *ICDE*, 2004.

[6] V. A. Ubhaya. Isotone optimization I. *Approx. Theory*, 12:146–159, 1974.

[7] D. Šuc and I. Bratko. Induction of qualitative tree. In *Proc. of European Conference of Machine Learning in 2001 (LNCS 2167)*, pages 442–453. Springer, 2001.

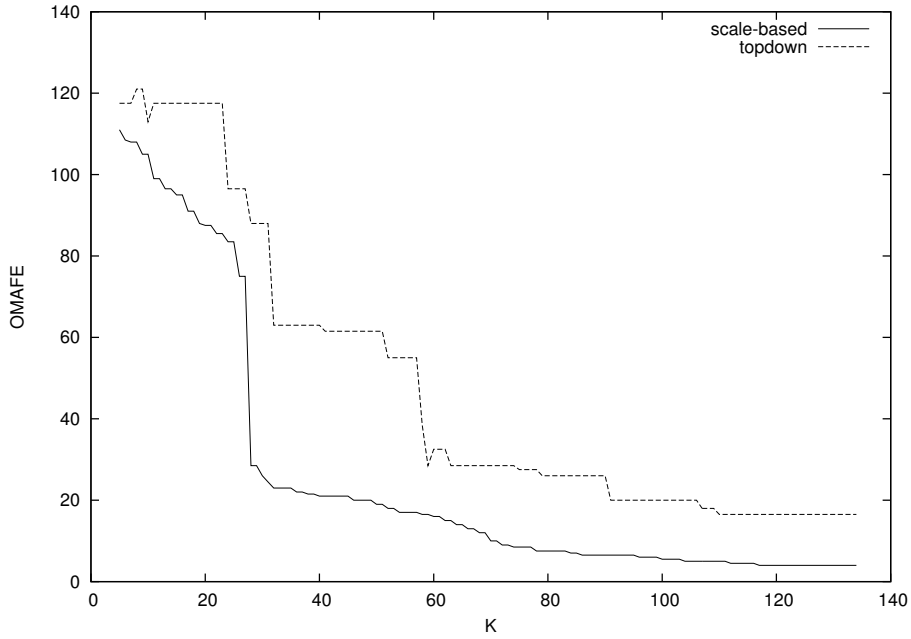

Figure 5. OMAFE (error measure) versus number of segments K for ECG data for the two algorithms: top-down linear approximation and our scale-based algorithm. Lower is better.