Mathematical Programming Algorithms for Regression-based Nonlinear Filtering in $I\!R^N$

by N.D. Sidiropoulos and R. Bro

T.R. 97-26

Mathematical Programming Algorithms for Regression-based Nonlinear Filtering in $\mathbb{R}^N$

N.D. Sidiropoulos and R. Bro

Abstract

Constrained regression problems appear in the context of optimal nonlinear filtering, as well as in a variety of other contexts, e.g., chromatographic analysis in chemometrics and manufacturing, and spectral estimation. This paper presents novel mathematical programming algorithms for some important constrained regression problems in $\mathbb{R}^N$. For brevity, we focus on four key problems, namely, locally monotonic regression (the optimal counterpart of iterated median filtering), and the related problem of piecewise monotonic regression, runlength-constrained regression (a useful segmentation and edge detection technique), and uni- and oligo-modal regression (of interest in chromatography and spectral estimation). The proposed algorithms are *exact* and *efficient*, and they also naturally suggest slightly suboptimal but very fast approximate algorithms, which may be preferable in practice.

N.D. Sidiropoulos is with the Institute for Systems Research, University of Maryland, College Park, MD 20742 U.S.A. He can be reached at (301) 405-7411, or via e-mail at nikos@glue.umd.edu

R. Bro is with the Chemometrics Group, Food Technology, The Royal Vet. & Agri. University, Rolighedsvej 30, DK-1958 Frederiksberg C, Denmark. He can be reached via e-mail at rb@kvl.dk

I. Introduction

For the purposes of this paper, a constrained regression problem is an optimization problem of the following form: Given a vector[1] $\mathbf{y} \in I\!\!R^N$, find $\mathbf{x}$ to

$$\textbf{minimize}: d(\mathbf{y} - \mathbf{x})$$

$$\textbf{subject to}: \mathbf{x} \in \Phi$$

where $d(\cdot)$ is typically some metric or semi-metric, and Φ is some set of feasible (or, *admissible*) solutions. When Φ is such that the fact that the constraints imposed by Φ are nowhere *locally* violated implies membership in Φ, then we shall say that we have a locally constrained regression problem. Furthermore, if $d(\cdot)$ is squared ℓ_2 norm, then similarity is measured via Euclidean distance, and we have a Least Squares (LS) regression problem.

LS regression is important for several reasons. First, LS regression is optimal (in the Maximum Likelihood sense) when measurement errors are additive Gaussian. Gaussianity is an assumption that is most often made in practice, for both practical (tractability) and theoretical (Central Limit Theorem [1]) considerations. Least squares regression has been applied to a wide variety of problems [2].

Euclidean distance is measured via the ℓ_2 norm. Other norms are also of interest, e.g., absolute distance is measured via the ℓ_1 norm. If $d(\cdot)$ is the ℓ_1 norm, we have a Least Absolute Error (LAE) regression problem. It is well known [3] that the use of the ℓ_1 norm instead of the ℓ_2 norm adds a certain measure of robustness to the regression, and this is especially important in nonlinear filtering applications. Least Absolute Error regression is optimal (in the Maximum Likelihood sense) when measurement errors are additive and Laplacian-distributed. The Laplacian is a much longer tailed distribution than the Gaussian. In this paper, we consider both LS and LAE regression, under various constraint sets Φ.

Another issue is whether or not Φ allows the elements of $\mathbf{x}$ to take on continuous or only discrete (and finitely many) values. While the latter is often the case of interest in digital filtering applications (and has been considered in detail by Sidiropoulos in [4], [5], [6]), in many other applications (e.g., chromatography) one is interested in regression in $I\!\!R^N$ [7].

Locally constrained regression problems find application in numerous diverse disciplines, including nonlinear filtering and segmentation [8], [9], [10], [4], [5], [6], statistics [11], psychology

[1]In this paper, we use the terms *vector* and (finite) *sequence* interchangeably

[12], databases [13], biology [14], texture perception [15], optimum decoding for magnetic media storage [16], and holographic data storage [17]. In the context of nonlinear filtering, constrained regression offers optimal counterparts of standard nonlinear filters, such as the iterated median, or morphological filters like opening and closing, while obviating the restriction to monotone increasing operators [4], [5].

In this paper, we consider *locally monotonic regression* and the related problem of *piecewise monotonic regression*, *runlength-constrained regression*, and *uni- and oligo-modal regression* in $\mathbb{R}^N$, which find application in nonlinear filtering, segmentation and edge detection, chromatography and spectral estimation. The first three are locally constrained problems, whereas uni- and oligo-modal regression is not.

A. Organization

The rest of this paper is structured as follows. Section II presents a general quantization result for a class of LAE regressions.

Section III addresses the locally monotonic regression problem in $\mathbb{R}^N$, and includes subsections on background and motivation (III-A), previous approaches (III-B), locally monotonic LAE regression (III-C), monotone regression and Kruskal's algorithm (III-D), and locally monotonic LS regression (III-E).

Section IV defines, motivates, and solves the problem of piecewise monotonic regression in $\mathbb{R}^N$, using a novel hybrid algorithm based on Dynamic Programming (DP) and an enhanced variant of Kruskal's algorithm; this is presented in subsection IV-B. This Section also includes subsections on necessary DP background (IV-A), and complexity analysis of the proposed algorithm (IV-C).

Section V presents detailed simulation experiments on locally monotonic (subsection V-A) and piecewise monotonic (subsection V-B) LS regression.

Section VI is concerned with runlength-constrained regression in $\mathbb{R}^N$, and discusses a novel pure DP algorithm for solving it. It includes subsections on complexity (VI-A), a fast sub-optimal counterpart (VI-B), and a discussion on how to modify the algorithm to accommodate a LAE metric (VI-C).

Section VII considers the problem of oligo-modal regression in $\mathbb{R}^N$, and includes a subsection on oligo-modal LAE regression (subsection VII-A), unimodal LS regression (subsection VII-B), and oligo-modal LS regression (subsection VII-C).

Finally, in Section VIII we draw conclusions and point to further research directions.

II. A General Quantization Result for a Class of LAE Regressions

Following [9], given any $\mathbf{x} \in \mathbb{R}^N$, we define its associated *sign skeleton* $\mathbf{s_x} \in \{-1, 0, 1\}^{N-1}$ as follows:

$$s_{\mathbf{x}}(n) = \begin{cases} -1 & if\ x(n+1) < x(n) \\ +1 & if\ x(n+1) > x(n) \\ 0 & if\ x(n+1) = x(n) \end{cases}$$

We have the following Lemma, which is a direct generalization of Lemma 4 of [9].

Lemma 1: Suppose that $d(\cdot)$ is an ℓ_p norm and $p \in [1, \infty)$, i.e.,

$$d(\mathbf{y} - \mathbf{x}) = \left(\sum_{n=1}^{N} |y(n) - x(n)|^p \right)^{\frac{1}{p}}$$

for some $p \in [1, \infty)$. Furthermore, suppose that membership of $\mathbf{x}$ in Φ can be determined by sole knowledge of its sign skeleton $\mathbf{s_x}$. If $\mathbf{x}$ is a solution of

$$\textbf{minimize} : d(\mathbf{y} - \mathbf{x})$$

$$\textbf{subject to} : \mathbf{x} \in \Phi$$

then $\mathbf{x}$ is a piecewise-constant sequence whose pieces are constant regressions of the corresponding segments of $\mathbf{y}$ under the given distance metric.

Proof: Lemma 4 of [9] makes the same claim for the special case of locally monotonic regression. The proof follows along the lines of proof in [9]. In particular, the assumption that membership of $\mathbf{x}$ in Φ can be determined by sole knowledge of its sign skeleton $\mathbf{s_x}$ means that there exists some $\Theta(\Phi) \subset \{-1, 0, 1\}^{N-1}$ such that $\mathbf{x} \in \Phi$ if and only if $\mathbf{s_x} \in \Theta(\Phi)$, in which case

$$min_{\mathbf{x} \in \Phi}\ d(\mathbf{y} - \mathbf{x}) =$$

$$min_{\mathbf{s_x} \in \Theta(\Phi)} \left[min_{\mathbf{x} \mid \mathbf{s_x}}\ d(\mathbf{y} - \mathbf{x}) \right]$$

Fix $\mathbf{s_x}$, and consider the inner minimization. The optimal $\mathbf{x}$ can always be thought of as a piecewise constant sequence (some or all of its pieces may contain just one element). If any given constant piece of $\mathbf{x}$ is not a constant regression of the corresponding elements of $\mathbf{y}$, then its fit (and thus the fit of $\mathbf{x}$) may always be improved by perturbing its level by an infinitesimal amount to bring it a bit closer to the said regression *without changing the sign skeleton* $\mathbf{s_x}$. This is true because strict inequality allows for an open ball of free movement of level in either direction.

This contradicts conditional optimality of $\mathbf{x}$ conditioned on $\mathbf{s_x}$. Thus the optimal $\mathbf{x}$ for any given $\mathbf{s_x}$ is a piecewise-constant sequence whose pieces are constant regressions of the corresponding segments of $\mathbf{y}$ under the given distance metric. This holds for all $\mathbf{s_x}$, and the result follows. ■

Lemma 2: If $d(\cdot)$ is ℓ_1 norm, i.e., $d(\mathbf{y}-\mathbf{x}) = \sum_{n=1}^{N} |y(n) - x(n)|$, and membership of $\mathbf{x}$ in Φ can be determined by sole knowledge of its sign skeleton $\mathbf{s_x}$, then it holds that if $\mathbf{x}$ is a solution of

$$\textbf{minimize}: d(\mathbf{y}-\mathbf{x})$$

$$\textbf{subject to}: \mathbf{x} \in \Phi$$

then for all $n = 1, 2, \cdots, N$, there exists an $m \in \{1, 2, \cdots, N\}$ such that $x(n) = y(m)$.

Proof: Constant regression under ℓ_1 amounts to picking the median of the elements involved (if the number of elements is even, any of the medians would do just as well); e.g., cf. [9]. In any case, the selected median is one of the above elements. ■

Corollary 1: Suppose that $d(\cdot)$ is ℓ_1 norm. Define

$$\mathcal{A} = \{v \in \mathbb{R} \mid \exists\, n \in \{1, 2, \cdots, N\} \; : \; y(n) = v\}$$

Furthermore, suppose that Φ is such that membership of $\mathbf{x}$ in Φ can be determined by sole knowledge of its sign skeleton $\mathbf{s_x}$. Then, $\mathbf{x}$ is a solution of

$$\textbf{minimize}: d(\mathbf{y}-\mathbf{x})$$

$$\textbf{subject to}: \mathbf{x} \in \Phi$$

if and only if $\mathbf{x}$ is a solution of

$$\textbf{minimize}: d(\mathbf{y}-\mathbf{x})$$

$$\textbf{subject to}: \mathbf{x} \in \Phi \cap \mathcal{A}^N$$

This is significant, because *it reduces regression over a subset of $\mathbb{R}^N$ to a finite problem.* In all cases of interest to us, Φ is neither convex nor does it have the algebraic structure of a subspace, thus the original problem is a difficult one. The finite problem, on the other hand, often admits efficient algorithmic solution. In addition, reduction to a finite problem immediately implies existence of a solution to the original problem.

In the sequel, we explore several regression problems that satisfy the conditions of this result. Actually, *all* LAE versions of nonlinear regressions considered herein fall under this category, and their digital counterparts admit efficient solution. As we will see, the corresponding LS versions do not admit such a "universal" solution, and have to be addressed on a one-by-one basis using different means.

III. Locally Monotonic Regression in $\mathbb{R}^N$

A. Background on Locally Monotonic Regression

Locally monotonic regression is the optimal counterpart of iterated median filtering. In [9], Restrepo and Bovik developed an elegant mathematical framework in which they studied locally monotonic regressions in $\mathbf{R}^N$. They proved existence of such regressions, and provided algorithms for their computation; however, their algorithms were *very* complex (actually, of complexity exponential in N, the size of the input sample). In addition to existence, they were able to show that locally monotonic regression admits a Maximum Likelihood interpretation [10].

In [5], Sidiropoulos considered *digital* locally monotonic regressions, in which the output symbols are drawn from a finite alphabet, and, by making a connection to Viterbi decoding, provided a very fast *linear in N* algorithm that computes any such regression, be it under a metric, semi-metric, or arbitrary bounded per-letter cost measure.

Before we proceed, we need some definitions.

If $\mathbf{x}$ is a real-valued sequence (string) of length N, and γ is any integer less than or equal to N, then a *segment* of $\mathbf{x}$ of length γ is any substring of γ consecutive components of $\mathbf{x}$. Let $\mathbf{x}_i^{i+\gamma-1} = \{x(i), \cdots, x(i+\gamma-1)\}$, $i \geq 0$, $i+\gamma \leq N$, be any such segment. $\mathbf{x}_i^{i+\gamma-1}$ is monotonic if either $x(i) \leq x(i+1) \leq \cdots \leq x(i+\gamma-1)$, or $x(i) \geq x(i+1) \geq \cdots \geq x(i+\gamma-1)$.

Definition 1: A real-valued sequence, $\mathbf{x}$, of length N, is *locally monotonic* of degree $\alpha \leq N$ (or *lomo-α*, or simply *lomo* in case α is understood) if each and every one of its segments of length α is monotonic.

Notice that some segments may be increasing, others decreasing, and the sequence may still pass the test of local monotonicity. In general, monotonicity implies local monotonicity, but not vice-versa. Also, any sequence is locally monotonic of degree $\alpha = 2$, so the interesting degrees are 3 through N. Throughout the following we assume that $3 \leq \alpha \leq N$. If $\alpha \leq \beta \leq N$, then a sequence of length N that is lomo-β is lomo-α as well; thus, the *lomotonicity* of a sequence is defined as the highest degree of local monotonicity that it possesses [9].

A sequence $\mathbf{x}$ is said to exhibit an increasing (resp. decreasing) transition at coordinate i if

$x(i) < x(i+1)$ (resp. $x(i) > x(i+1)$). The following (cf. [18], [9], [19]) is a key property of locally monotonic signals: If $\mathbf{x}$ is locally monotonic of degree α, then $\mathbf{x}$ has a constant segment (run of identical symbols) of length at least $\alpha - 1$ in between an increasing and a decreasing transition. The reverse is also true.

The study of local monotonicity (which led to the idea of locally monotonic regression) has a relatively long history in the field of nonlinear filtering. Local monotonicity appeared in the study of the set of root signals of the median filter [18], [19], [20], [21], [22], [23], [24]. The median is arguably the most widely known and used nonlinear filter. As it turns out, in 1-D, the set of root signals (fixed points) of a median filter of a certain length (i.e., the set of those signals that are not affected at all as they pass through the given median filter) is the set of locally monotonic signals of a certain degree of local monotonicity. In other words, the class of locally monotonic signals of a given degree of local monotonicity is the set of roots of a median filter of a certain length.

In the sense that, at least in 1-D and modulo some pathological cases, iterations of the median do converge to a median root that somehow resembles the original input signal, and median roots are locally monotonic signals, one may think of iterated median filtering as an attempt to converge to a locally monotonic signal that approximates the original input signal. However, this resemblance cannot be *a priori* quantified, and the final output may be well off the original input in certain cases.

It is then natural to think about asking for the *best* possible locally monotonic approximation of the input in hand, rather than settle for the arbitrarily chosen coarse locally monotonic replica provided by iterated median filtering. This gives rise to locally monotonic regression, which can deliver significant gains in terms of distortion [9], [10], [5]. Here, goodness of fit is usually measured in terms of a metric or semi-metric [9], or likelihood [10]; or, it can be an arbitrary bounded per-letter cost measure [5].

B. Previous Approaches

Previous algorithms for locally monotonic regression include:

- The original *Tube* and *Blotching* algorithms of Restrepo and Bovik [9]; these are of exponential complexity (see also [10] for some properties of locally monotonic regression).
- The work of the first author of this paper, on fast *digital* locally monotonic regression [5]. This algorithm has complexity *linear* in the number of samples, and is therefore very efficient.

However, it solves a *digital* (i.e., *finite alphabet*) problem. By proper choice of quantization, this may well be entirely satisfactory in certain applications, yet unsatisfactory in others, where one needs very fine granularity in the solution.

- The recent work of de la Vega and Restrepo [25] on an improved algorithm for locally monotonic regression in $\mathbb{R}^N$. The algorithm given in [25] applies to the case of $\alpha = 3$ (the smallest meaningful α), and, according to the authors, it is of polynomial complexity. Computational complexity, and generalization of the algorithm to the case of arbitrary α are open questions, according to the authors of the paper, as of this writing.

None of the above efficiently solves the general locally monotonic regression problem in $\mathbb{R}^N$. In the sequel, we present an exact fast algorithm for locally monotonic LAE regression in $\mathbb{R}^N$, and an improved fast algorithm for locally monotonic LS pseudo-regression in $\mathbb{R}^N$.

C. Locally Monotonic LAE Regression in $\mathbb{R}^N$

Consider the locally monotonic LAE regression problem:

$$\textbf{minimize}: ||\mathbf{y} - \mathbf{x}||_1$$

$$\textbf{subject to}: \mathbf{x} \in \Lambda_\alpha^N$$

where Λ_α^N is the set of all sequences of N elements of $\mathbb{R}$ which are locally monotonic of lomo-degree α. It is easy to see that Λ_α^N satisfies the condition of Corollary 1. Define $\mathcal{A}$ as in Corollary 1, and consider the resulting digital problem:

$$\textbf{minimize}: ||\mathbf{y} - \mathbf{x}||_1$$

$$\textbf{subject to}: \mathbf{x} \in \Lambda_\alpha^N \cap \mathcal{A}^N$$

This is an instance of digital locally monotonic regression, which has been solved in [5] by means of DP. The complexity of the digital algorithm is $O(|\mathcal{A}|^2\alpha N)$; $\mathcal{A}$ as defined in Corollary 1 has at most N elements; thus the computational complexity of locally monotonic LAE regression in $\mathbb{R}^N$ is $O(N^3\alpha)$. This solution is *exact* and *efficient*, although it may be quite tedious for long observation sequences.

One may improve upon complexity by capitalizing on the following observation. Since at each point the value of the optimum solution is necessarily the median of a small collection of inputs in the neighborhood of the given point, one may restrict the alphabet at time n to be the set of input values in a suitable neighborhood about n. The longest streaks in a locally monotonic regression of degree α are usually bounded in length by $k \times \alpha$, where k is a small integer constant; one may therefore pick a *local alphabet* for $x(n)$ that consists of, say, at most 4α elements, and run DP under this restriction. This will be beneficial for α small relative to N, and will lead to complexity $O(\alpha^3 N)$, without any significant loss in performance.

D. Additional Background - Monotone Regression

In order to properly understand *locally* monotonic LS regression, one needs, as can be expected, to develop a good understanding of *monotone* LS regression.

Monotone (say, increasing) LS regression is the following problem: Given $\mathbf{y}$, find $\mathbf{x}$ to

$$\textbf{minimize}: ||\mathbf{y} - \mathbf{x}||_2^2$$

$$\textbf{subject to}: \mathbf{x} \textit{ is monotone increasing}$$

or, to be more precise, *monotone non-decreasing*, i.e., letting $x(n)$ denote the n-th element of the N-dimensional vector (sequence of N elements) $\mathbf{x}$:

$$x(1) \leq x(2) \leq \cdots \leq x(N)$$

This problem is of interest in statistics (e.g., cf. [11]), nonlinear filtering (e.g., cf. [8]), and various other disciplines, including psychology [12], databases [13], biology [14], and texture perception [15].

In the latter applications, the interest in monotone regression is *indirect*; e.g., it is a substep in algorithms for so-called *multi-dimensional scaling*, a technique that allows trend spotting and correlation analysis in very large datasets.

There exist various ways of solving a monotone regression problem. The obvious observation is that monotone regression is a special case of Quadratic Programming (QP), and, therefore, it can be solved using one's favorite QP routine. However, this approach is not very efficient (theoretically $O(N^{3.5})$ using the state-of-art interior point methods of QP [26]; in practice, $O(N^4)$ using common routines for large datasets), and it leaves much to be desired, especially in view of the fact that monotone regression is a *very special* QP program that exhibits a lot of potentially exploitable *structure*.

Indeed better ways of solving it exist. In 1964, Kruskal [12] came up with what appears to be the best way of attacking this problem, as a by-product of his pioneering work in multidimensional scaling (see also [27]). Kruskal's approach is an *iterative* one, that makes use of a simple but clever observation to reduce the problem of monotone least squares regression to a proper finite sequence of averaging steps. The theoretical complexity of Kruskal's algorithm is (rather loosely) upper bounded by $O(N^2)$; in practice, it is almost $O(N)$.

For our purposes, as we will soon see, we will need a fast algorithm for a slightly different problem. In particular, we will need an algorithm for the following equality-constrained non-decreasing problem:

$$\textbf{minimize}: ||\mathbf{y}-\mathbf{x}||_2^2$$

$$\textbf{subject to}: x(1) \leq x(2) \leq \cdots \leq x(N-(\alpha-1)) \leq$$

$$\leq x(N-(\alpha-2)) = x(N-(\alpha-1)) = \cdots = x(N)$$

i.e., the last $\alpha - 1$ elements of the regression should be equal, for some $3 \leq \alpha \leq N$. In addition, we will need a fast algorithm for the problem:

$$\textbf{minimize}: ||\mathbf{y}-\mathbf{x}||_2^2$$

$$\textbf{subject to}: x(1) \geq x(2) \geq \cdots \geq x(N-(\alpha-1)) \geq$$

$$\geq x(N-(\alpha-2)) = x(N-(\alpha-1)) = \cdots = x(N)$$

i.e., the corresponding non-increasing problem. We will refer to both these problems as *suffix-run-constrained monotone regression.* Clearly, if one has an algorithm that solves one of them, one may also solve the other. In particular, suppose that we have an algorithm that solves the following *prefix-run-constrained monotone non-decreasing regression* problem:

$$\textbf{minimize}: ||\mathbf{y}-\mathbf{x}||_2^2$$

$$\textbf{subject to}: x(1) = x(2) = x(\alpha-1) \leq$$

$$\leq x(\alpha) \leq \cdots \leq x(N)$$

then, it is easy to see that the first of the former two suffix-run-terminated monotone regression problems can be solved by feeding $-rev(\mathbf{y})$ (where the $rev(\cdot)$ operation simply reverses the order of elements of its argument) as input to the said algorithm, and then computing $-rev(\mathbf{x})$; whereas the second of the former two problems can be solved by feeding $rev(\mathbf{y})$ to the said algorithm, and then computing $rev(\mathbf{x})$. These statements can be easily verified mathematically, using only the fact that $||\mathbf{y}-\mathbf{x}||_2^2 = ||rev(\mathbf{y}) - rev(\mathbf{x})||_2^2$, and $||\mathbf{x}||_2^2 = ||-\mathbf{x}||_2^2$.

In addition, it will be beneficial to be able to compute all sub-regressions i.e., in the context of the prefix-run-constrained monotone non-decreasing regression algorithm predicated above, it will be useful to be able to obtain, as by-products, all prefix-run-constrained sub-regressions on the first k elements of $\mathbf{y}$ for $k \geq \alpha - 1$.

Such an algorithm is presented in the Appendix (in which $\mathbf{b}$ plays the role of $\mathbf{x}$). It is based on Kruskal's original monotone regression algorithm, but (i) incorporates the prefix constraint, and (ii) produces as a "free" by-product all such sub-regressions. Given that Kruskal's algorithm is provably *correct* [27], the proof of correctness of this modified algorithm is straightforward.

E. Locally Monotonic LS Regression in $\mathbb{R}^N$

Let us now consider locally monotonic LS regression:

$$\textbf{minimize}: ||\mathbf{y} - \mathbf{x}||_2^2$$

$$\textbf{subject to}: \mathbf{x} \in \Lambda_\alpha^N$$

This problem turns out to be more difficult than its LAE counterpart. In particular, we cannot resort to an equivalent of Corollary 1, for one does not really exist: the set of all averages of a finite collection of reals is finite but very big.

The idea is to first run a computationally cheap fast digital locally monotonic LS regression algorithm, as described in [5], to determine an approximate solution, then call the monotone regression algorithm given in the appendix *just once* for each monotone piece of this digital interim solution to improve its fit, and bring it closer to the optimum real-valued regression. If the resulting real-valued suffix-terminated monotone regression piece respects local endpoint consistency constraints (e.g., that the first element of an increasing segment following a decreasing segment should be greater than the last element of the decreasing segment), then inserting it in place of the corresponding digital piece will improve the fit without violating the local monotonicity constraints. The process is repeated for the remaining pieces in a sequential fashion.

Since the fast digital algorithm is *linear* in N (but quadratic in the size of the digital alphabet - so very fine discretization can be costly), and the monotone regression algorithm given in the appendix is better than quadratic in N, it follows that the complexity of this sub-optimum two-step process is loosely bounded above by $O(N^2)$ and is much better than this figure in practice.

Through simulation, one may easily convince oneself that the performance degradation caused by this sub-optimum two-step approach relative to the true regression in $\mathbb{R}^N$ is usually small, given a proper choice of quantization levels for the discrete part, while the computational gains are very significant. Thus, this two-step approach would normally be the method of choice in practice.

IV. Piecewise Monotonic LS Regression in $\mathbb{R}^N$

Piecewise monotonic LS regression of degree α is defined as the following problem:

$$\textbf{minimize}: ||\mathbf{y}-\mathbf{x}||_2^2$$

$$\textbf{subject to}: \mathbf{x} \in \Pi_\alpha^N$$

where Π_α^N is the set of all sequences of N elements of $\mathbb{R}$ which can be constructed by concatenating monotone pieces, each one of which can be *either* increasing *or* decreasing, *and* of length at least $\alpha - 1$. If $\mathbf{x} \in \Pi_\alpha^N$, then we shall say that $\mathbf{x}$ is piecewise monotonic of pimo-degree α. Note that for $\beta > \alpha$, any $\mathbf{x}$ that is pimo-β is pimo-α as well; thus this regression is defined over a family of nested approximation "spaces", and the resulting optimum fit is a non-decreasing function of pimo-degree, α.

Observe that Π_α^N satisfies the condition of Corollary 1. In addition, the digital counterpart of this problem admits efficient solution via DP, in a manner very similar to [5]. Therefore the LAE variant of the above problem in $\mathbb{R}^N$ can be solved by means of a suitable digital algorithm, at a complexity cost of $O(N^3\alpha)$, as predicated by Corollary 1. Here we attack the LS problem.

There are several reasons for introducing piecewise monotonic regression. First, observe that $\Lambda_\alpha^N \subset \Pi_\alpha^N$, i.e., every locally monotonic sequence of degree of local monotonicity α is a piecewise monotonic sequence of degree of piecewise monotonicity α, but the reverse is not true. Thus, piecewise monotonic regression can be seen as regression over a feasible set that includes all locally monotonic sequences. However, it is still small enough to retain the attractive nonlinear filtering properties of locally monotonic regression, and, actually, improve on some deficiencies of the latter.

From an optimization viewpoint, piecewise monotonic regression relieves some of the constraints associated with local monotonicity. In particular, it relieves the segment endpoint consistency constraints (e.g., that the first element of an increasing segment following a decreasing segment should be greater than the last element of the decreasing segment), as well as the constraint that individual monotone segments should terminate with a suffix string of at least $\alpha - 1$ *equal* elements. These latter constraints have been inspired by the properties exhibited by the roots of the median filter. The suffix constraint, in particular, leads to a now well-understood undesirable effect of median filtering known as *streaking* [28]. The idea is that by relaxing the suffix constraint one may be able to reduce[2] streaking without sacrificing the noise suppression and edge preservation capabilities of the resulting nonlinear regression-based filter. This is indeed the case, as we will soon see by means of simulation.

Also important is the fact that relaxing some of the local monotonicity constraints effectively *decouples* the problem in a particular sense, and allows us to develop a fast hybrid DP - Kruskal algorithm for piecewise monotonic LS regression, a development that is not possible for locally monotonic LS regression, due to an intricate coupling between pieces of the solution for the latter problem. Of course, locally monotonic LAE regression in $\mathbb{R}^N$ is entirely tractable, and high-quality approximate efficient solutions for locally monotonic LS regression are also available, as discussed earlier.

Our primary aim here is to introduce a novel related nonlinear edge-preserving smoothing technique, that may improve upon the properties of locally monotonic LS regression *and* admits efficient exact solution in $\mathbb{R}^N$.

Finally, observe that since $\Lambda_\alpha^N \subset \Pi_\alpha^N$, i.e., every locally monotonic sequence of degree of local monotonicity α is a piecewise monotonic sequence of degree of piecewise monotonicity α, it follows that piecewise monotonic LS regression provides a lower bound on the LS fit achievable by locally monotonic LS regression of the same α.

A. Additional Background - Dynamic Programming and the Viterbi Algorithm

In the sequel, we will need to invoke the Viterbi Algorithm (VA) in a "master" mode that will control an underlying special-purpose Quadratic Program in a "slave" mode to solve the piecewise monotonic regression problem in $\mathbb{R}^N$. For this purpose, at this point it is useful to clarify exactly what kind of optimization problem may be solved by the VA.

The VA is the name by which many engineers refer to an instance of forward DP. In a nutshell, the VA is nothing but a clever method to search for an N-tuple $\{s(n)\}_{n=0}^{N-1}$ of variables (each one of which can take on only a finite number of values) that minimizes:

$$\sum_{n=0}^{N-1} c_n(s(n), s(n-1))$$

[2]Streaking can be also attributed in part to the use of an ℓ_p norm, $p = 1, 2$, and cannot be avoided altogether; cf. Corollary 1.

where $s(-1)$ is a (given) dummy variable, $c_n(\cdot,\cdot)$ is some arbitrary one-step transition cost, and the $s(n)$'s are thought of as *state variables*, i.e., variables that summarize the past of a system in so far as its future evolution is concerned. Without loss of generality we may assume that all state variables take on values in the same alphabet, $\mathcal{A}$. The VA avoids exhaustive search and brings the complexity of this minimization from a brute-force $O(|\mathcal{A}|^N)$ down to a reasonable $O(|\mathcal{A}|^2 N)$ in the *worst case*; actual complexity depends on the given $c_n(\cdot,\cdot)$'s.

The VA achieves substantial computational savings by capitalizing on a simple observation: by taking the last summation term out of the sum, and conditioning on any given choice of the $s(N-2)$ state variable, the resulting two terms of the cost functional can be minimized independently; this effectively decouples the problem and results in two independent problems, one of which can be trivially solved. By iterating this argument backwards in time, one may realize significant computational gains.

This is a very powerful and pervasive optimization technique [29], and it finds application in many diverse areas, including, but not limited to:

- Optimum decoding of convolutional codes [30], [31], [32].
- Speech and character recognition [33].
- State estimation in Hidden Markov Models (HMM's) [33].
- Optimal Control [34].
- Game theory [34].

In the context of solving constrained optimization problems, the key to successfully using the VA is to come up with a *state* of partial solutions with respect to the given set of constraints [4], [5], [6].

B. Hybrid Programming Algorithm for Piecewise Monotonic LS Regression

Consider the piecewise monotonic least squares regression problem:

$$\textbf{minimize}: ||\mathbf{y}-\mathbf{x}||_2^2$$

$$\textbf{subject to}: \mathbf{x} \in \Pi_\alpha^N$$

where Π_α^N is the set of all sequences of N elements of $\mathbb{R}$ which are piecewise monotonic of pimo-degree α.

For each element $\mathbf{x}$ of Π_α^N, define its associated *trend switching set*, $T(\mathbf{x})$, as the set of indices

$\{t_1, t_2, \cdots, t_K\}$ where $\mathbf{x}$ exhibits a trend transition (increasing to decreasing, or vice-versa)[3]. Let us assume, without loss of generality, that $\mathbf{x}$ always starts with an increasing trend[4].

For any $\mathbf{x} \in \Pi_\alpha^N$, $T(\mathbf{x})$ has the following properties, whose proofs are elementary:

Property 1: For $\mathbf{x} \in \Pi_\alpha^N$, $K = |T(\mathbf{x})| \leq trunc(\frac{N}{\alpha-1})$.

Convention 1: By convention, we allow the elements of $T(\mathbf{x})$ to take on the value N (which serves as a "gate" for unused trend switches); this allows us to fix the size of $T(\mathbf{x})$.

Property 2: Let $\mathbf{x} \in \Pi_\alpha^N$, and $T(\mathbf{x}) = \{t_1, t_2, \cdots, t_K\}$. Then, obviously,

$$\alpha - 1 \leq t_1 \leq t_2 \leq \cdots \leq t_K \leq N$$

Property 3: If both t_{i+1} and t_i are not equal to N, then $t_{i+1} - t_i \geq \alpha - 1$, $\forall i = 1, 2, \cdots, K-1$.

Property 4: If $t_i = N$, then $t_j = N$, $\forall i < j \leq K$.

Property 5: In between $[t_{i-1}+1, t_i]$, $\mathbf{x}$ is constrained to be monotonic (specifically: non-decreasing for i: odd; non-increasing for i: even); otherwise, it is unrestricted.

Proposition 1: Under Convention 1, $\mathbf{x} \in \Pi_\alpha^N$, if and only if $T(\mathbf{x})$ satisfies Properties 1, 2, 3, 4, 5.

Let us denote by V_α^N the set of all trend switching sets (i.e., sets of positive integer K-tuples) that satisfy Properties 1, 2, 3, 4, 5. By the above proposition, $\mathbf{x} \in \Pi_\alpha^N$ if and only if $T(\mathbf{x}) \in V_\alpha^N$. Therefore, the piecewise monotonic least squares regression problem:

$$\textbf{minimize}: ||\mathbf{y} - \mathbf{x}||_2^2$$

$$\textbf{subject to}: \mathbf{x} \in \Pi_\alpha^N$$

is equivalent to:

$$\textbf{minimize}: ||\mathbf{y} - \mathbf{x}||_2^2$$

[3]One may determine the trend switching set by knowledge of the sign skeleton but not vice-versa. In the case of piecewise monotone regression, feasibility of a candidate solution may be determined from knowledge of its sign skeleton or its trend switching set. The latter leads to a more compact parameterization of the feasible set, and thus to more efficient algorithms; it is therefore adopted here as the parameterization of choice.

[4]Otherwise, we simply run the process twice, one time starting with an increasing trend, the other starting with a decreasing trend, and pick the best of the two; this will double the amount of computation, but will not affect the *order* of computational complexity.

$$\textbf{subject to} : T(\mathbf{x}) \in V_\alpha^N$$

Define

$$\mathcal{J}(\mathbf{x}) = ||\mathbf{y} - \mathbf{x}||_2^2$$

and the problem becomes:

$$min_{\mathbf{x}|T(\mathbf{x}) \in V_\alpha^N} \mathcal{J}(\mathbf{x})$$

$$= min_{\{t_1,t_2,\cdots,t_K\} \in V_\alpha^N} \{min \mathcal{J}(\mathbf{x}|T(\mathbf{x}) = \{t_1, t_2, \cdots, t_K\})\}$$

$$= min_{\{t_1,t_2,\cdots,t_K\} \in V_\alpha^N} \sum_{i=1}^{K} min \mathcal{E}(t_{i-1}, t_i)$$

$$= min_{\{t_1,t_2,\cdots,t_K\} \in V_\alpha^N} \sum_{i=1}^{K} \mathcal{E}^*(t_{i-1}, t_i)$$

where the last two steps follow from Property 5, $t_0 = 0$ by convention, and $\mathcal{E}^*(t_{i-1}, t_i)$ is the cost (fit) of an optimal monotonic (specifically: non-decreasing for i: odd; non-increasing for i: even) least squares regression on the elements of $\mathbf{y}$ between indices $t_{i-1} + 1$ and t_i inclusive.

Now the problem is in a form suitable for Dynamic Programming. In particular, the problem can be solved by DP over the trend switching variables, t_i, and, as we will see shortly in the complexity analysis below, the required aggregate trellis computations involve the solution of a very special QP program (namely, monotone regression) for each node at each stage in the trellis.

Note that the hybrid DP-QP approach proposed herein is very different from the one proposed in [5], which does not carry over in $\mathbb{R}^N$. The present approach is related, in spirit, to a DP algorithm of Blake for the solution of a nonlinear regularization problem [35], and a similar DP program of Bellman for solving a piecewise linear approximation problem subject to a budget on the number of pieces that one may use to construct the optimum approximation [36]. Both these DP algorithms do not need to repeatedly invoke a sophisticated optimization subroutine to perform their respective tasks; in a sense, their respective problems are easier than piecewise monotonic LS regression.

C. Complexity of Novel Algorithm for Piecewise Monotonic Least Squares Regression in $\mathbb{R}^N$

Consider Figure 1, which depicts the nodes (states) at stage-i and their respective potential predecessors at stage-$(i-1)$. At stage-i, the node tags correspond to all the possible values of t_i, and similarly for stage-$(i-1)$. At stage-i, the bottom node has just one predecessor that is α nodes apart. The next node going upwards has just two predecessors, the furthest of which is

$\alpha + 1$ nodes apart. A generic node, n, of stage-i has $n - i(\alpha - 1) + 1$ predecessors, the furthest of which is $n - (i-1)(\alpha - 1) + 1$ apart. The top node (N) is $N - (i-1)(\alpha-1) + 1$ apart from its furthest predecessor.

Each node at stage-i is visited in turn, and a decision is made as to which of the associated potential predecessors is best for the node in hand. To do this, one needs to calculate $\mathcal{E}^*(t_{i-1}, t_i)$ for the specific value of t_i assigned to the node in hand, and *all* values of t_{i-1} assigned to its potential predecessor nodes, add the respective results to the corresponding cumulative costs of the potential predecessor nodes, pick the one that gives minimum error, update the cumulative cost of the node in hand, set up a pointer to its best predecessor, then move on to the next node, the next stage, and so on and so forth.

The *crucial observation* is that a "generic" node, say the one that has been assigned the value $t_i = n$, only needs to make *one* call to Kruskal's monotone regression algorithm, and this call suffices to compute *all* node transition cost figures $\mathcal{E}^*(t_{i-1}, n)$ for all potential predecessors. Indeed, according to our earlier discussion, node n only needs to call the algorithm once with the longest input, and all required sub-regressions will be computed along the way for free. The length of this longest input is $n - (i-1)(\alpha-1) + 1$.

Since Kruskal's algorithm is better than quadratic, it follows that the computational cost for all required computations for the worst (top) node at stage-i is bounded above by

$$O\left([N - (i-1)(\alpha-1) + 1]^2\right)$$

Stage-i has a total of $[N - i(\alpha-1) + 1]$ nodes; thus the computational cost for all required computations for stage-i is (quite loosely) bounded above by

$$O\left([N - i(\alpha-1) + 1] \times [N - (i-1)(\alpha-1) + 1]^2\right)$$

There exist at most $\frac{N}{\alpha-1}$ stages, and the worst stage is $i = 1$, so the total computational cost for the entire regression is (again, rather loosely) bounded above by

$$O\left(\frac{N}{\alpha-1}\,[N - (\alpha-1) + 1]\,(N+1)^2\right)$$

which is bounded above by

$$O\left(\frac{(N+1)^3}{\alpha-1}\,[(N+1) - (\alpha-1)]\right)$$

We emphasize that this bound is loose, i.e., actual complexity is usually much less than the order given by the bound. In practice, the bound may be conservative by an order of magnitude. However, the given bound seems to be hard to improve for arbitrary α (more sophisticated counting arguments result in a bound of the same order).

V. Simulation

The purpose of this section is to enhance the reader's intuition by providing and discussing the results of several simulation experiments on locally monotonic and piecewise monotonic LS regression in $\mathbb{R}^N$.

As we have seen, the corresponding LAE versions of these problems may be reduced to suitable digital problems which admit efficient solution. For manuscript length considerations, here we do not present simulation results for the LAE versions, and refer the reader to [4], [5], [6].

A. Locally Monotonic LS Regression in $\mathbb{R}^N$

At the conceptual level, we now have three ways to approach the problem of locally constrained LS regression in $\mathbb{R}^N$ for arbitrary α. These are (in order of decreasing complexity) exact algorithms [9], the two-step algorithm, and the digital algorithm. All three provide *feasible* solutions from the constraint set. What differentiates these algorithms is the trade-off that each achieves between LS fit and computational complexity.

In terms of complexity, the digital algorithm is the simplest; the two-step algorithm is a very close second; and there exists a very significant gap between the two-step algorithm and exact algorithms. Actually, exact algorithms are not an option for a modest $N = 300$ samples, due to exponential complexity explosion.

The two-step algorithm, as we shall see in this section, may improve the quality of the digital solution by a considerable amount in terms of LS fit.

The two efficient algorithms have been implemented in C and $MATLAB^{(c)}$. Figures 2 through 19 present the results of these experiments. Numerical results are summarized in Table I. The input, $\mathbf{y}$, is a 300-point ($N = 300$) sample of a noisy version of an ECG signal taken from the signal processing information base (`http://www.spib.rice.edu`). In all figures, the spike train at the bottom depicts optimum trend switches, as detected by the digital algorithm. In the captions, $|\mathcal{A}|$ is the size of the digital alphabet, and α is lomo-degree.

Figures 2-4 present the results for $|\mathcal{A}| = 40$, $\alpha = 50$. In particular, Figure 2 presents the input versus the two-step real-valued regression, for $|\mathcal{A}| = 40$, $\alpha = 50$. Figure 3 presents the input versus the digital regression, for the same $|\mathcal{A}|$, α. Figure 4 compares the above two-step real-valued regression and the corresponding digital regression.

Figures 5-7 present the same for $|\mathcal{A}| = 40$, $\alpha = 20$; subsequent triples of Figures do the same for various values of $|\mathcal{A}|$, α.

Let us consider the numerical results summarized in Table I. An important general observation is that the difference in runtime between the digital and the two-step real-valued fast suboptimal algorithm is really negligible. In absolute terms, both run in the order of a few seconds, or a couple of minutes, at worst.

In terms of fit, the two-step real-valued algorithm always improves upon the fit of the digital algorithm, sometimes by as much as thirty percent. This improvement becomes more pronounced, as expected, when $|\mathcal{A}|$ is reduced (which also leads to a rapid decrease in complexity, as expected, since the digital algorithm is quadratic in $|\mathcal{A}|$). Furthermore, the percent improvement in fit afforded by the two-step real-valued algorithm increases with decreasing α. This may be intuitively explained as follows. When α is small, the digital algorithm is forced to closely track the input, thereby exhibiting tracking oscillations, in a manner similar to a phenomenon known as *slope over/under load* in delta modulation [37].

From Table I, we may also observe that the second step of the two-step real-valued algorithm is able to compensate for inaccuracies caused by using a small alphabet in the first (digital) step. Note that the fit of the two-step algorithm remains essentially the same regardless of whether $|\mathcal{A}|$ is 40 or 20, at least for $\alpha = 50$, $\alpha = 20$, with a small degradation when $\alpha = 5$.

These observations are also summarized in Figure 20(a). Figure 20(b) depicts CPU time as a function of α and $|\mathcal{A}|$. Note that actual time complexity scales up somewhat better than theoretically expected.

B. Piecewise Monotonic LS Regression in $\mathbb{R}^N$

The results of experiments on piecewise monotonic LS Regression in $\mathbb{R}^N$ are presented in Figures 21,22(a)-(b), and Table II.

Figure 21 presents the result of piecewise monotonic regression of pimo-degree $\alpha = 75$ for the noisy $N = 300$-point ECG signal. Figures 22(a)-(b) present the same for $\alpha = 50$, and $\alpha = 20$, respectively. In all three Figures, the spike train at the bottom depicts the locations of optimal trend switches, as detected by the exact hybrid algorithm. Table II summarizes LS fit and CPU time for these three regressions.

A couple of important remarks are in order. First, one may verify that streaking is much less pronounced compared to locally monotonic regression, e.g., compare the corresponding regressions for $\alpha = 20$: the first pulse is blotched by locally monotonic regression, yet it is well-preserved by piecewise monotonic regression. Second, this reduction in streaking is not at the expense of noise smoothing and/or the ability to follow signal edges in the data, at least for moderate values of α. The drawback is that, as one may easily convince oneself, for very small α piecewise monotonic LS regression is more susceptible to outliers than locally monotonic LS regression.

The choice of ECG signal data was not arbitrary; aside from it exhibiting sharp level transitions (thus being a natural candidate for the application of novel nonlinear smoothing techniques), this choice was also meant to hint on an interesting application of piecewise monotonic and locally monotonic regression. In the interpretation of ECG signals, certain features like the relative timing and duration of the so-called P-wave, QRS-complex, and T-wave are very informative. These features are relatively easily picked up by a trained eye, even when the data is noisy, yet automatic detection and segmentation is difficult, due to changes in heart rate, noise, and other considerations [38].

Consider Figure 23. It depicts another portion of the same ECG, and the result of piecewise monotonic regression of degree $\alpha = 20$. The P-wave, QRS-complex, and T-wave have been manually annotated, and the spike train at the bottom depicts the locations of optimal trend switches, as detected by the exact hybrid algorithm. Notice that P,Q,R,S, and T can be accurately localized by looking at the detected optimal trend switches: this spike train may be used to develop an automated feature extraction / ECG segmentation process that does not rely heavily on heuristics and is robust in the presense of noise.

VI. Runlength-Constrained Least Squares Regression in $\mathbb{R}^N$

Let us now consider the following runlength-constrained least squares regression problem:

$$\textbf{minimize}: ||\mathbf{y} - \mathbf{x}||_2^2$$

$$\textbf{subject to}: \mathbf{x} \in P_M^N$$

where P_M^N is the set of all sequences of N elements of $\mathbb{R}$ which are piecewise-constant, and the length of constituent pieces is bounded below by M.

This problem appears in the context of segmentation and edge detection. Its digital (finite-alphabet) version has been considered in [4], where an efficient DP algorithm was developed for its solution, and the properties of the associated I/O operator were investigated. Here, we are interested in the same problem, but this time in $\mathbb{R}^N$.

One may mimic the development presented for the case of piecewise monotonic regression, and associate to each element $\mathbf{x}$ of P_M^N its corresponding *level switching set* $L(\mathbf{x})$. M plays the role of $\alpha - 1$, and the rest of the derivation remains the same.

A major simplification in the case of runlength-constrained regression relative to piecewise monotonic regression, is that the former does not require any relatively sophisticated regression subroutine to be invoked repeatedly by the master trellis computation: given two subsequent level switches under consideration, the optimum constant LS regression in-between them is simply the average of the corresponding input elements. What is more, all constant sub-regressions can be computed on-the-fly, while computing the longest constant regression that is required for a given state in the trellis. The resulting program is pure DP.

A. Complexity of Novel Algorithm for Runlength-Constrained Least Squares Regression in $\mathbb{R}^N$

One may pre-compute all required averages and associated costs in-between any two indices at a cost of $O(N^2)$, *before* running the DP program. These pre-computed data can be stored in a table for easy access during runtime. Now replace $\alpha - 1$ by M in the complexity analysis of the proposed algorithm for piecewise monotonic regression. Since all averages and associated costs are stored in the table, $\mathcal{E}^*(t_{i-1}, t_i)$ is readily available for any t_i and t_{i-1}; thus the computational cost for all required computations for the worst (top) node at stage-i is bounded above by $O(N - (i-1)M + 1)$. Stage-i has a total of $(N - iM + 1)$ nodes; thus the computational cost for all required computations for stage-i is (quite tightly) bounded above by $O([N - iM + 1] \times [N - (i-1)M + 1])$; There exist at most $\frac{N}{M}$ stages, and the worst stage is $i = 1$, so the total computational cost for the entire regression is bounded above by

$$O\left(\frac{N}{M}[N - M + 1] \times [N + 1]\right)$$

Observe that when $M = N$ this bound predicts complexity $O(N)$, which is exactly what is needed for computing the average of N elements.

B. Very Fast Sub-Optimal Algorithm

Again, the above algorithm naturally suggests a very fast sub-optimal counterpart. The idea is to first run a computationally cheap fast digital runlength-constrained regression algorithm, as described in [4], to determine almost-optimum level switches, then compute the required averages in-between adjacent level switches, to determine the optimum real-valued regression *given the level switches determined using the fast digital algorithm.* Since the former fast digital algorithm is *linear* in N, and the latter step is trivial, it follows that the complexity of this sub-optimum two-step process is bounded above by $O(N)$. Of course, if the number of levels utilized for the discrete part is large, it may pay to use the direct algorithm, and this may happen in practice, since the complexity of the direct method is not prohibitive.

C. The LAE case

Even though one may invoke Corollary 1 to obtain an exact $O(N^3M)$ algorithm for runlength-constrained LAE regression in $\mathbb{R}^N$, it turns out that there also exists an alternative way of achieving the same goal.

Rather than invoking Corollary 1, one may proceed as follows. The above runlength-constrained LS regression in $\mathbb{R}^N$ algorithm can be modified to handle runlength-constrained LAE regression in $\mathbb{R}^N$, by replacing the averaging operation in-between level switches by a median calculation. This is true by virtue of the fact that, given n numbers, their median is the constant that minimizes the sum of absolute errors between itself and all n given numbers (e.g., cf. [9]). Again, computational savings may be realized by using an efficient running median algorithm, e.g., the one of Huang *et al* [39] based on histogram updates, for computing all required constant sub-regressions under a least absolute error measure in-between level switches.

Other types of locally constrained (e.g., piecewise-convex) regression problems can be handled in the same spirit.

VII. Oligo-modal Regression in $\mathbb{R}^N$

Oligo-modal regression is the following problem:

$$\textbf{minimize}: d(\mathbf{y}-\mathbf{x})$$

$$\textbf{subject to}: \pi(\mathbf{x}) = P$$

where $\pi(\mathbf{x})$ is the number of peaks in $\mathbf{x}$, and P is some constant (note that a *plateau* is counted as a single peak). This problem is of interest in e.g., chromatography [7], and special problems in spectral estimation [33].

A. Oligo-modal LAE Regression in $\mathbb{R}^N$

Observe that one may determine whether or not $\pi(\mathbf{x}) = P$ by sole knowledge of the sign skeleton of $\mathbf{x}$. In addition, one may construct a DP program to solve a discretized finite-alphabet version of a given oligo-modal regression problem. This can be done along the lines of [4], [5], [40], or as explained in [7] for an alternative approach to unimodal regression. One may show that the complexity of this program will be $O(|\mathcal{A}|^2PN)$, where $|\mathcal{A}|$ is the size of the finite alphabet, P is the number of peaks, and N is the total length of the regression. It therefore follows from Corollary 1 that, provided $d(\cdot)$ is ℓ_1 norm, one may construct a suitable DP program to solve the oligo-modal LAE regression in $\mathbb{R}^N$ at a complexity cost of $O(N^3P)$.

B. Unimodal LS Regression in $\mathbb{R}^N$

Unimodal regression is the following problem. Given $\mathbf{y} \in \mathbb{R}^N$, find $\mathbf{x} \in \mathbb{R}^N$ to:

$$\textbf{minimize}: ||\mathbf{y} - \mathbf{x}||_2^2$$

$$\textbf{subject to}: \mathbf{x}: \textit{unimodal}$$

i.e., $\pi(\mathbf{x}) = 1$, $\mathbf{x}$ has only one peak. We are particularly interested in non-negative unimodal regression (note that a bounded problem can always be transformed to a non-negative problem), in which case the unimodality constraint can be expressed as follows:

$$x(0) \geq 0; \; x(N) \geq 0;$$

$$x(n) \geq x(n-1), \; n = 2, \cdots, j;$$

$$x(n) \leq x(n-1), \; n = j+1, \cdots, N = size(\mathbf{x})$$

for some mode location, j, which *is itself subject to optimization.*

If one fixes the mode location, the constrained problem is an instance of QP. Exhaustive search through all N possible mode locations, each time solving a QP program, gives a first naive way of solving it.

This problem is of interest in, among other things, chromatographic analysis and flow injection analysis [7]. It has been considered in detail in Bro & Sidiropoulos [7]. A fundamental and surprising result of [7] is that unimodal least squares regression (including optimization of mode location) is not anymore difficult than two simple Kruskal monotone regressions. This is well under $O(N^2)$ and, in practice, almost $O(N)$; a marked improvement over the naive exhaustive QP-based approach, which amounts to $O(N^{4.5})$-$O(N^5)$, depending on the QP routine used.

C. Oligo-modal LS Regression in $\mathbb{R}^N$

Along the lines of the previously presented hybrid algorithm for piecewise monotonic LS regression, one may envision the construction of a similar algorithm for oligo-modal LS regression, consisting of a master DP module, and a Kruskal monotone regression (or, our own unimodal regression) module. However, it turns out that even after fixing peak and valley locations, the resulting problem is *not* decomposable in a series of either monotone or unimodal independent sub-regressions in-between these locations, as shown by means of counter-example in [7]. Thus DP won't help solve this LS problem *exactly*. The point we would like to make is that it may help solve the problem *approximately*.

An alternative *approximate* approach would be as follows. First, one may construct a DP program to solve a discretized finite-alphabet version of a given oligo-modal LS regression problem. This program is a straightforward extension of the unimodal program given in [7]. Note that the resulting DP program provides an *exact* solution of the finite-alphabet problem, which is also an *approximate* solution of the $\mathbb{R}^N$ problem. As mentioned earlier, the complexity of this program will be $O(|\mathcal{A}|^2PN)$, where $|\mathcal{A}|$ is the size of the finite alphabet, P is the number of peaks, and N is the total length of the regression. Depending on $\mathcal{A}$, this solution may or may not be appropriate, both in terms of precision and in terms of complexity. In any case, it will be a sub-optimum solution of the $\mathbb{R}^N$ problem, since it solves the oligo-modal regression problem *under the extra constraint that the output regression levels may only assume values in a fixed finite set*, so it provides a feasible solution whose fit is generally worse than that of a true regression in $\mathbb{R}^N$.

There's no obvious way to further reduce complexity, but there's a straightforward way to improve accuracy. That is, once the discrete solution is found, as above, one may further improve it by posing and solving a QP program to compute the optimum regression *conditioned* on the detected locations of peaks and valleys. This may improve the fit, at a computational expense of $O(N^4)$ for typical QP routines. Observe that QP *will only be called once* here, and its complexity is independent of the number of peaks, so for moderate N this is feasible.

VIII. Conclusions

We have presented algorithms for several interesting constrained regression problems in $\mathbb{R}^N$. A general quantization result for a class of LAE regressions in $\mathbb{R}^N$ sets the stage for the efficient solution of many regression problems in this class, by means of DP solutions of suitable *finite* problems. The scope of this result extends well beyond the collection of problems considered herein.

Locally monotonic regression has been considered in detail, as it is of special importance to the nonlinear filtering community. Piecewise monotonic regression is a closely related problem, which naturally suggests itself. The algorithm for piecewise monotonic LS regression in $\mathbb{R}^N$ is especially interesting, as it invloves an innovative *master-slave* combination of DP and an enhanced version of a very special (yet relatively little appreciated) efficient algorithm for monotone regression, due to Kruskal.

The development of the novel algorithm for runlength-constrained regression followed from the above. Finally, the idea of using Kruskal's monotone regression algorithm came to us while working on unimodal regression, and we have presented exact and approximate algorithms for oligo-modal LAE and LS regression, respectively.

This work has many interesting ramifications, several of which are currently under investigation. Future work includes the study of stability and convergence of multi-objective algorithms incorporating some of the present algorithms as sub-steps; and extensions to handle additional prior knowledge, e.g., smoothness or equality constraints.

Related $MATLAB^{(c)}$ and C-code can be found in `http://www.glue.umd.edu/ nikos`; also at `http://newton.foodsci.kvl.dk/rasmus.html`.

IX. Acknowledgments

The work of the first author has been supported in part by core funds from the NSF ERC program, made available through the Communications and Signal Processing Group of the Institute for Systems Research of the University of Maryland; and industry, through Martin-Marietta Chair in Systems Engineering funds. His work was motivated in part by earlier work of Al Bovik and Alfredo (Palacios) Restrepo. The work of the second author was supported in part by the Nordic Industry Foundation project P93149, and the $F\emptyset TEK$ Foundation.

X. Appendix - MATLAB code for Prefix-constrained Monotone Regression in $\mathbb{R}^N$

```
function [b,B,AllBs]=modRasmusmonreg(y,alpha);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A variant of monotonic Least Squares regression according                 %
% to J. B. Kruskal, 1964.                                                    %
% Extensions to handle calculation of sub-regressions by Rasmus Bro          %
% Minor modification to handle the equality constr. by R. Bro & N. Sidiropoulos%
% b     = min||y-b||_2 subj to monotone increase, AND first alpha-1 els equal %
% B     = b, but condensed (runlength-encoded)                               %
% AllBs = All sub-regressions, i.e. AllBs(1:i,i) is the                     %
%         regression of y(1:i) under the above constraints.                  %
% NOTE: of course, alpha-1 <= size(y), and the first alpha-2 columns of      %
% the matrix AllBs are not of interest, and should be considered invalid     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extensions and modification:    R. Bro and N. Sidiropoulos, 1997          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note that a monotone increasing regression on y that terminates with      %
% alpha-1 equal symbols can be performed by feeding - rev(y) as input       %
% to this routine, and then -rev(b), where rev() reverses the order of      %
% the elements of its argument. Similarly, a monotone decreasing             %
% regression on y that terminates with                                       %
% alpha-1 equal symbols can be performed by feeding  rev(y) as input        %
% to this routine, and then rev(b). These statements are easy to            %
% prove mathematically, using only the fact that || y - b||=||rev(x) - rev(b)||%
% and ||y|| = ||-x|| - N. Sidiropoulos & R. Bro                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

L = alpha - 1;
I=length(y);
ynew=zeros(I-L+1,1);
ynew(1,1) = mean(x(1:L,1));
ynew(2:I-L+1,1) = x(L+1:I,1);
id = ones(size(ynew));
id(1) = L;
B=[ynew id];

  AllBs=zeros(I,I);
  for j=1:L,
    AllBs(1:j,j) = ynew(1,1) * ones(j,1);
  end;

  i=1;
  while i<size(B,1)
    if B(i,1)>B(min(I,i+1),1)
       summ=B(i,2)+B(i+1,2);
       B=[B(1:i-1,:);[(B(i,1)*B(i,2)+B(i+1,1)*B(i+1,2))/(summ) summ];B(i+2:size(B,1),:)];
       OK=1;
       while OK
         if B(i,1)<B(max(1,i-1),1)
           summ=B(i,2)+B(i-1,2);
           B=[B(1:i-2,:);[(B(i,1)*B(i,2)+B(i-1,1)*B(i-1,2))/(summ) summ];B(i+1:size(B,1),:)];
           i=max(1,i-1);
         else
           OK=0;
         end
       end
```

```
        bInterim=[];
        for i2=1:i
           bInterim=[bInterim;zeros(B(i2,2),1)+B(i2,1)];
        end
        No=sum(B(1:i,2));
        AllBs(1:No,No)=bInterim;
    else
        i=i+1;
        bInterim=[];
        for i2=1:i
           bInterim=[bInterim;zeros(B(i2,2),1)+B(i2,1)];
        end
        No=sum(B(1:i,2));
        AllBs(1:No,No)=bInterim;
    end
end

b=[];
for i=1:size(B,1)
  b=[b;zeros(B(i,2),1)+B(i,1)];
end
```

References

[1] W. Feller, *An introduction to probability theory and its applications (Third edition - revised printing)*, Wiley, New York, 1970, Two volumes.

[2] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, New Jersey, 1996.

[3] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters: Principles and Applications*, Kluwer, Boston, 1990.

[4] N.D. Sidiropoulos, "The Viterbi Optimal Runlength-Constrained Approximation Nonlinear Filter", *IEEE Trans. Signal Processing*, vol. 44, no. 3, pp. 586–598, March 1996.

[5] N.D. Sidiropoulos, "Fast Digital Locally Monotonic Regression", *IEEE Trans. Signal Processing*, vol. 45, no. 2, Feb. 1997, (in press).

[6] N.D. Sidiropoulos, J.S. Baras, and C.A. Berenstein, "Weak Continuity with Structural Constraints", Submitted for publication, IEEE Trans. Signal Processing. Summary appears in Proc. 1996 IEEE Workshop on Statistical Signal and Array Processing, June 24-26, Corfu, Greece.

[7] R. Bro and N.D. Sidiropoulos, "Least Squares Algorithms Under Unimodality and Non-negativity Constraints", *Preprint*, 1997.

[8] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, Mass., 1987.

[9] A. Restrepo and A. C. Bovik, "Locally Monotonic Regression", *IEEE Trans. Signal Processing*, vol. 41, no. 9, pp. 2796–2810, Sep. 1993.

[10] A. Restrepo and A. C. Bovik, "Statistical Optimality of Locally Monotonic Regression", *IEEE Trans. Signal Processing*, vol. 42, pp. 1548–1550, Jun. 1994.

[11] R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk, *Statistical inference under order restrictions*, Wiley, New York, NY, 1972.

[12] J. B. Kruskal, "Nonmetric multidimensional scaling: a numerical method", *Psychometrika*, vol. 29, no. 2, pp. 115–129, June 1964.

[13] C. Faloutsos and King-Ip (David) Lin, "FastMap: A fast algorithm for indexing, data-mining, and visualization of traditional and multimedia datasets", *ACM SIGMOD*, May 1995, To appear; also available as ISR TR 94-80, CS-TR-3383, UMIACS-TR-94-132.

[14] David Sankoff and Joseph B. Kruskal, *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparisons*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1983.

[15] A. Ravishankar Rao and Jerry Lohse, "Identifying high level features of texture perception", in *SPIE Conference*, San Jose, Feb. 1992.

[16] B. Marcus, P Siegel, and J. Wolf, "Finite-state modulation codes for data storage", *IEEE J. on Selected Areas in Communications*, vol. 10, pp. 5–37, 1992.

[17] B. Marcus, ", Optimal receivers for 2-D runlength-constrained holographic data storage systems. Personal Communication, January 1996.

[18] S. G. Tyan, "Median filtering: deterministic properties", in *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*, T. S. Huang, Ed., pp. 197–217. Springer-Verlag, Berlin, 1981.

[19] N.C. Gallagher Jr. and G.W. Wise, "A theoretical analysis of the properties of median filters", *IEEE Trans. ASSP*, vol. ASSP-29, pp. 1136–1141, Dec. 1981.

[20] B. I. Justusson, "Median filtering: statistical properties", in *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*, T. S. Huang, Ed., pp. 161–196. Springer-Verlag, Berlin, 1981.

[21] T. A. Nodes and N. C. Gallagher, "Median filters: some modifications and their properties", *IEEE Trans. ASSP*, vol. ASSP-30, pp. 739–746, 1982.

[22] N.C. Gallagher Jr., "Median filters: a tutorial", in *Proc. IEEE Int. Symp. Circ., Syst., ISCAS-88*, 1988, pp. 1737–1744.

[23] A. C. Bovik, T. S. Huang, and D. C. Munson, "A generalization of median filtering using linear combinations of order statistics", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, pp. 1342–1349, 1983.

[24] A.C. Bovik, T.S. Huang, and D.C. Munson Jr., "The effect of median filtering on edge estimation and detection", *IEEE Trans. PAMI*, vol. PAMI-9, pp. 181–194, Mar. 1987.

[25] R. de la Vega and A. Restrepo, "Efficient Computation of Locally Monotonic Regression", *IEEE Signal Processing Letters*, vol. 3, no. 9, pp. 263–265, Sep. 1996.

[26] S-C. Fang and S. Puthenpura, *Linear Optimization and Extensions: Theory and Algorithms*, AT&T - Prentice-Hall, Englewood Cliffs, NJ, 1993.

[27] Jan De Leeuw, "Correctness of Kruskal's algorithms for monotone regression with ties", *Psychometrika*, vol. 42, no. 1, pp. 141–144, March 1977.

[28] A. C. Bovik, "Streaking in median filtered images", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, pp. 493–503, 1987.

[29] Hui-Ling Lou, "Implementing the Viterbi Algorithm", *IEEE Signal Processing Magazine*, vol. 12, no. 5, pp. 42–52, 1995.

[30] A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Trans. Information Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[31] J.K. Omura, "On the Viterbi decoding algorithm", *IEEE Trans. Information Theory*, vol. IT-15, pp. 177–179, Jan. 1969.

[32] B. Sklar, *Digital Communications*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[33] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[34] D. Bertsekas, *Dynamic Programming and Optimal Control, Vols. I and II*, Athena Scientific, Belmont, MA, 1995.

[35] A. Blake, "Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction", *IEEE Trans. PAMI*, vol. 11, no. 1, pp. 2–12, Jan. 1989.

[36] R. Bellman, "On the approximation of curves by line segments using dynamic programming", *Commun. ACM*, vol. 4, pp. 284, 1961.

[37] S. Haykin, *Communication Systems*, Wiley, NY, 3d edition, 1994.

[38] Dana H. Brooks and Robert S. MacLeod, "Electrical imaging of the heart", *IEEE Signal Processing Magazine*, vol. 14, no. 1, pp. 24–42, 1997.

[39] T.S. Huang, G.J. Yang, and G.Y. Tang, "A fast two-dimensional median filtering algorithm", *IEEE Trans. ASSP*, vol. ASSP-27, no. 1, pp. 13–18, 1979.

[40] N.D. Sidiropoulos and J. Baras, "On Adapting the Regularization Parameter of Weak Continuity and a Related Regression Problem", Tech. Rep. TR 97-8, Institute for Systems Research, University of Maryland, January 1997.

	digital, fit	2-step, fit	digital, cputime	2-step, cputime
$\|\mathcal{A}\| = 40,\ \alpha = 50$	6946	6910	1'45"	1'47"
$\|\mathcal{A}\| = 40,\ \alpha = 20$	1593	1518	40"	41"
$\|\mathcal{A}\| = 40,\ \alpha = 5$	447	358	15"	15.5"
$\|\mathcal{A}\| = 20,\ \alpha = 50$	6998	6912	30"	32"
$\|\mathcal{A}\| = 20,\ \alpha = 20$	1775	1518	15"	16"
$\|\mathcal{A}\| = 20,\ \alpha = 5$	640	417	5"	5.7"

TABLE I

FIT AND EXECUTION TIME COMPARISONS FOR LOCALLY MONOTONIC REGRESSION UNDER ℓ_2 (NETWORKED SUN SPARC10, $N = 300$ POINTS). THE DIGITAL PART HAS BEEN IMPLEMENTED IN C, WHEREAS THE SECOND STEP OF THE TWO-STEP ALGORITHM HAS BEEN IMPLEMENTED IN $MATLAB^{(R)}$. CPU TIME IS IN (MINUTES'SECONDS"), AND WAS REPORTED VIA A COMBINATION OF `cputime` AND UNIX `time`.

	fit	cputime
$\alpha = 75$	3539	12'20"
$\alpha = 50$	2206	17'11"
$\alpha = 20$	570	35'2"

TABLE II

FIT AND EXECUTION TIME FOR PIECEWISE MONOTONIC REGRESSION UNDER ℓ_2 (NETWORKED SUN SPARC10, $N = 300$ POINTS). THE HYBRID ALGORITHM HAS BEEN IMPLEMENTED IN COMPILED $MATLAB^{(R)}$. CPU TIME IS IN (MINUTES'SECONDS"), AND WAS REPORTED VIA `cputime`.

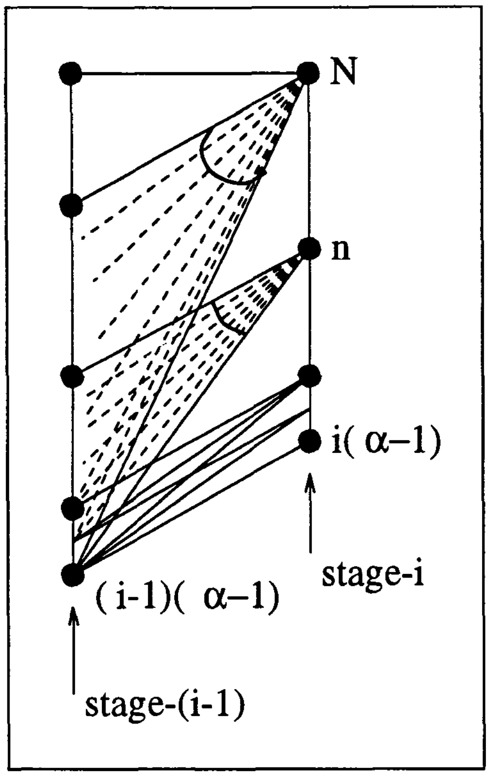


Fig. 1. Nodes (states) at stage-i and predecessors at stage-$(i-1)$. At stage-i, the bottom node has just one predecessor that is α nodes apart. The next node going upwards has just two predecessors, the furthest of which is $\alpha + 1$ nodes apart. A generic node, n, at stage-i has $n - i(\alpha - 1) + 1$ predecessors, the furthest of which is $n - (i-1)(\alpha - 1) + 1$ apart. The top node (N) is $N - (i-1)(\alpha - 1) + 1$ apart from its furthest predecessor. Note that node N is special, in the sense that it is possible to move from node N at stage-$(i-1)$ to node N at stage-i at zero cost.

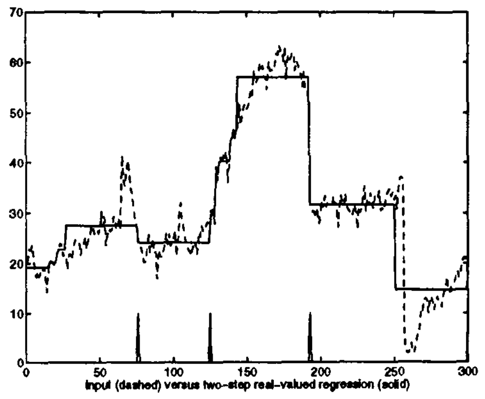


Fig. 2. Input (dashed) versus two-step real-valued (solid) regression, $|\mathcal{A}| = 40$, $\alpha = 50$.

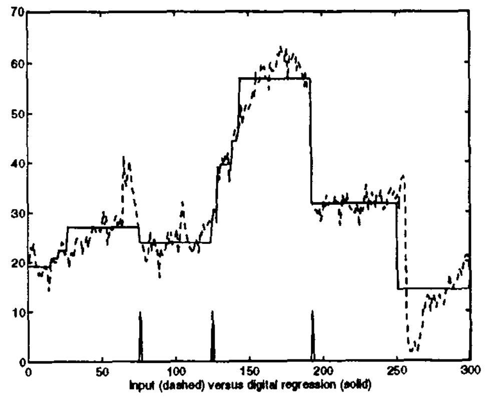


Fig. 3. Input (dashed) versus digital (solid) regression, $|\mathcal{A}| = 40$, $\alpha = 50$.

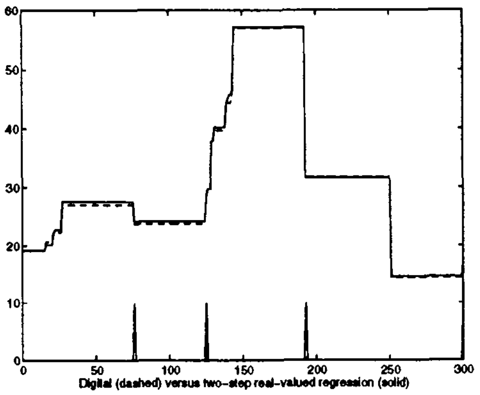


Fig. 4. Two-step real-valued (solid) versus digital (dashed) regression, $|\mathcal{A}| = 40$, $\alpha = 50$.

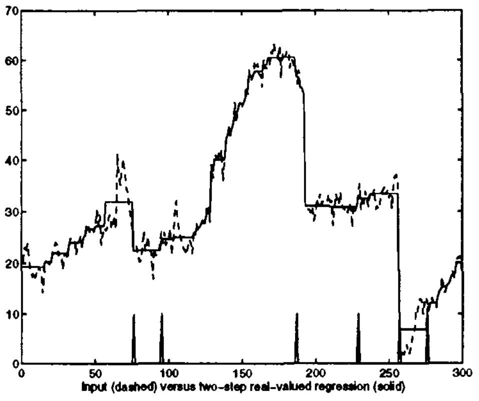


Fig. 5. Input (dashed) versus two-step real-valued (solid) regression, $|\mathcal{A}| = 40$, $\alpha = 20$.

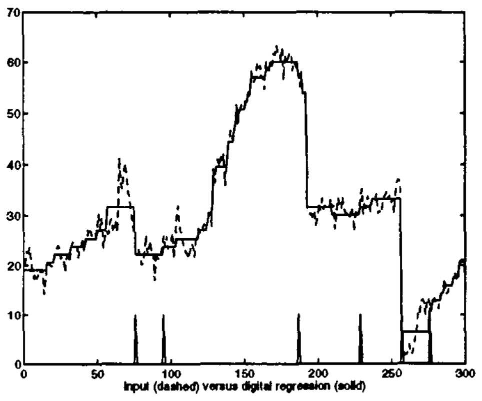


Fig. 6. Input (dashed) versus digital (solid) regression, $|\mathcal{A}| = 40$, $\alpha = 20$.

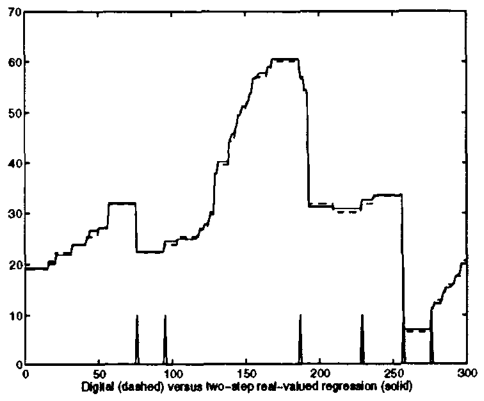


Fig. 7. Two-step real-valued (solid) versus digital (dashed) regression, $|\mathcal{A}| = 40$, $\alpha = 20$.

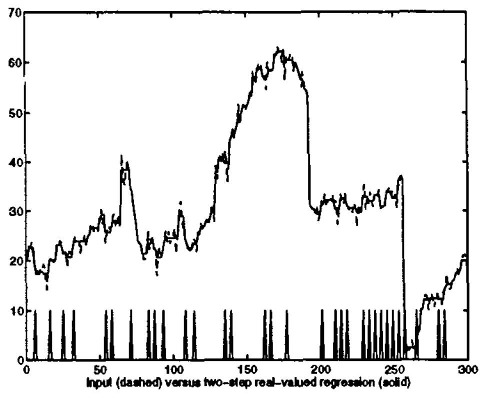


Fig. 8. Input (dashed) versus two-step real-valued (solid) regression, $|\mathcal{A}| = 40$, $\alpha = 5$.

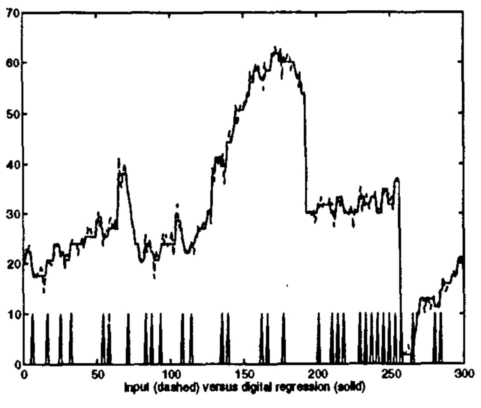


Fig. 9. Input (dashed) versus digital (solid) regression, $|\mathcal{A}| = 40$, $\alpha = 5$.

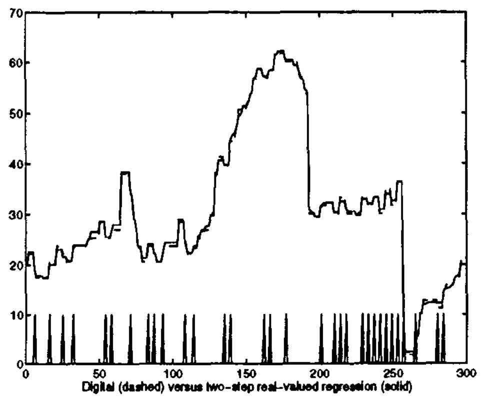


Fig. 10. Two-step real-valued (solid) versus digital (dashed) regression, $|\mathcal{A}| = 40$, $\alpha = 5$.

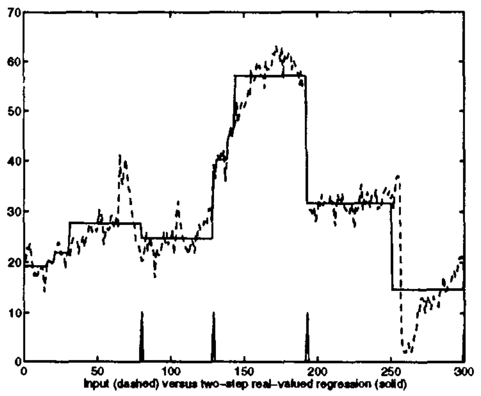


Fig. 11. Input (dashed) versus two-step real-valued (solid) regression, $|\mathcal{A}| = 20$, $\alpha = 50$.

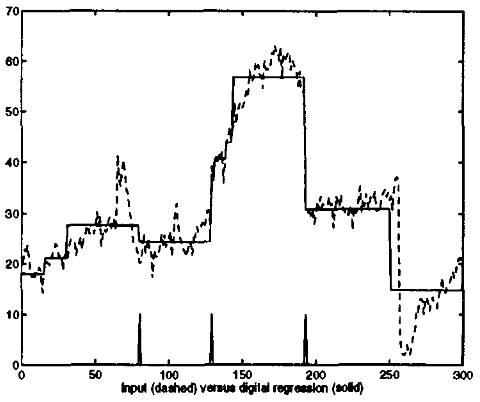


Fig. 12. Input (dashed) versus digital (solid) regression, $|\mathcal{A}| = 20$, $\alpha = 50$.

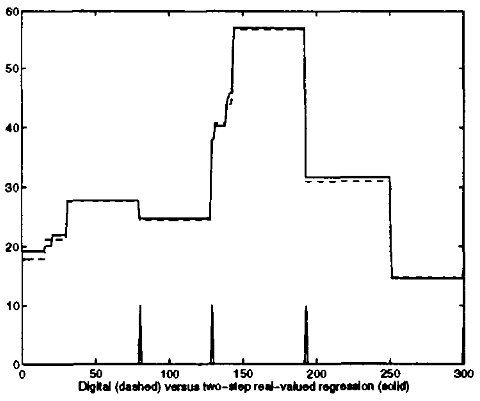


Fig. 13. Two-step real-valued (solid) versus digital (dashed) regression, $|\mathcal{A}| = 20$, $\alpha = 50$.

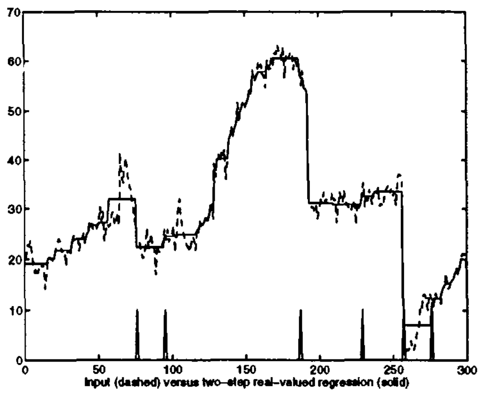


Fig. 14. Input (dashed) versus two-step real-valued (solid) regression, $|\mathcal{A}| = 20$, $\alpha = 20$.

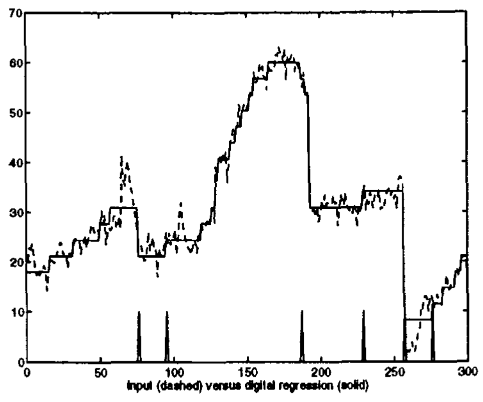


Fig. 15. Input (dashed) versus digital (solid) regression, $|\mathcal{A}| = 20$, $\alpha = 20$.

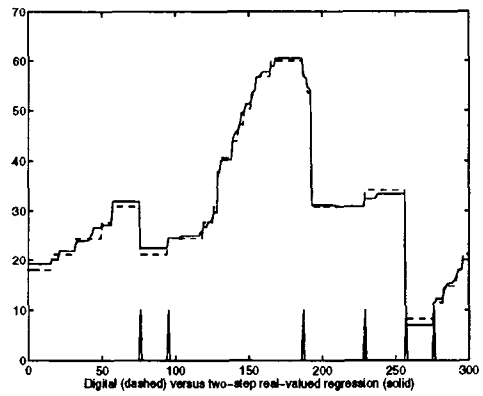


Fig. 16. Two-step real-valued (solid) versus digital (dashed) regression, $|\mathcal{A}| = 20$, $\alpha = 20$.

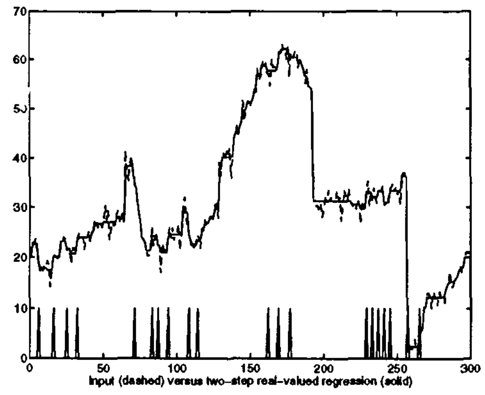


Fig. 17. Input (dashed) versus two-step real-valued (solid) regression, $|\mathcal{A}| = 20$, $\alpha = 5$.

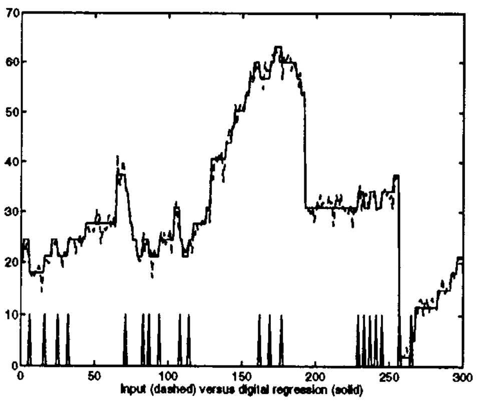


Fig. 18. Input (dashed) versus digital (solid) regression, $|\mathcal{A}| = 20$, $\alpha = 5$.

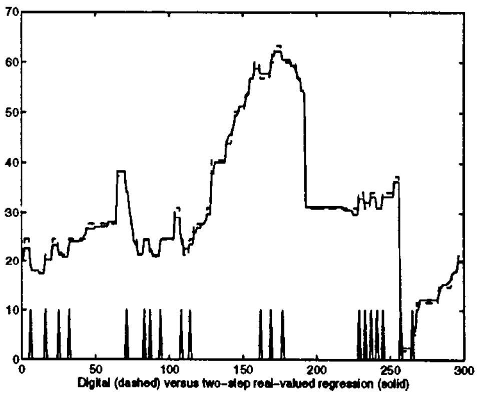
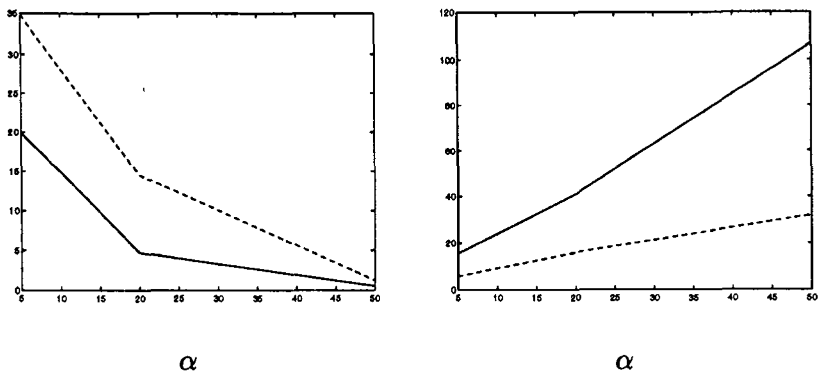


Fig. 19. Two-step real-valued (solid) versus digital(dashed) regression, $|\mathcal{A}| = 20$, $\alpha = 5$.



(a) Percent improvement in LS fit (b) CPU time (in seconds)

Fig. 20. (a) Percent improvement in LS fit curves as a function of α. Solid curve: $|\mathcal{A}| = 40$, dashed curve: $|\mathcal{A}| = 20$. This is the percent improvement afforded by the two-step algorithm versus the brute-force digital algorithm. (b) CPU time required by the two-step algorithm as a function of α. Solid curve: $|\mathcal{A}| = 40$, dashed curve: $|\mathcal{A}| = 20$. Note that complexity scales somewhat better than expected by the theoretical complexity analysis of the algorithm, i.e., actual complexity is sub-quadratic in $|\mathcal{A}|$.

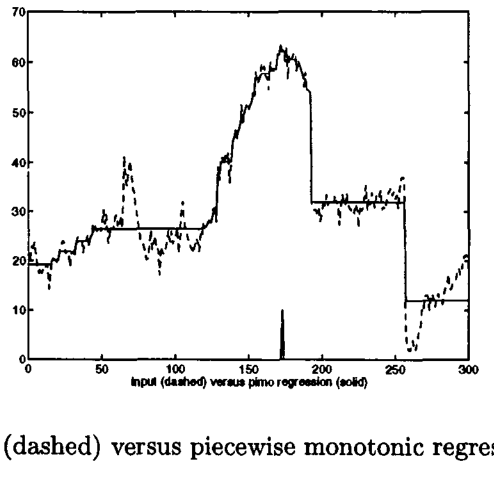
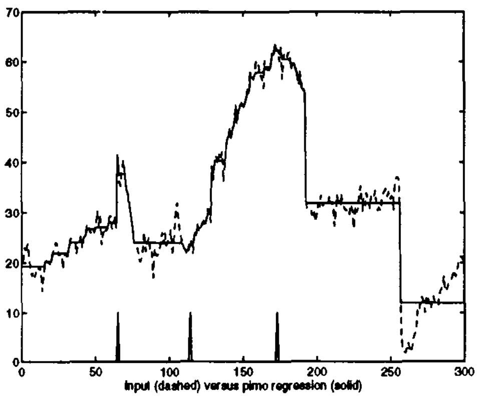
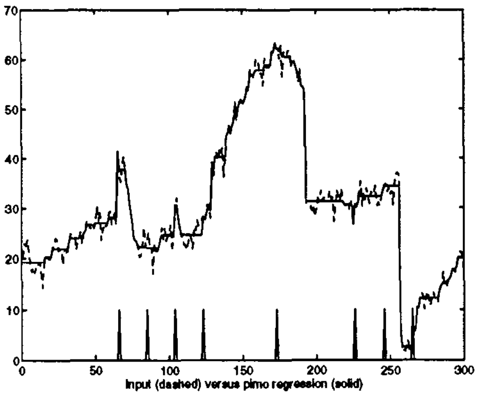


Fig. 21. Input (dashed) versus piecewise monotonic regression, $\alpha = 75$.



(a) $\alpha = 50$



(b) $\alpha = 20$

Fig. 22. Input (dashed) versus piecewise monotonic regressions for $\alpha = 50, 20$.

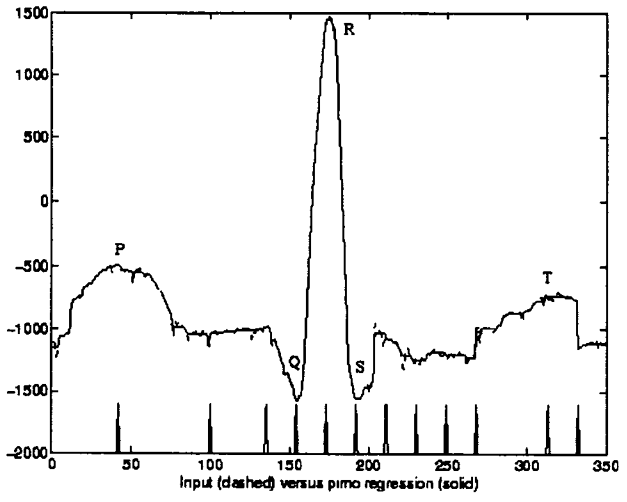


Fig. 23. Piecewise monotonic regression may aid in the detection of significant events in ECG signals. Here $\alpha = 20$.