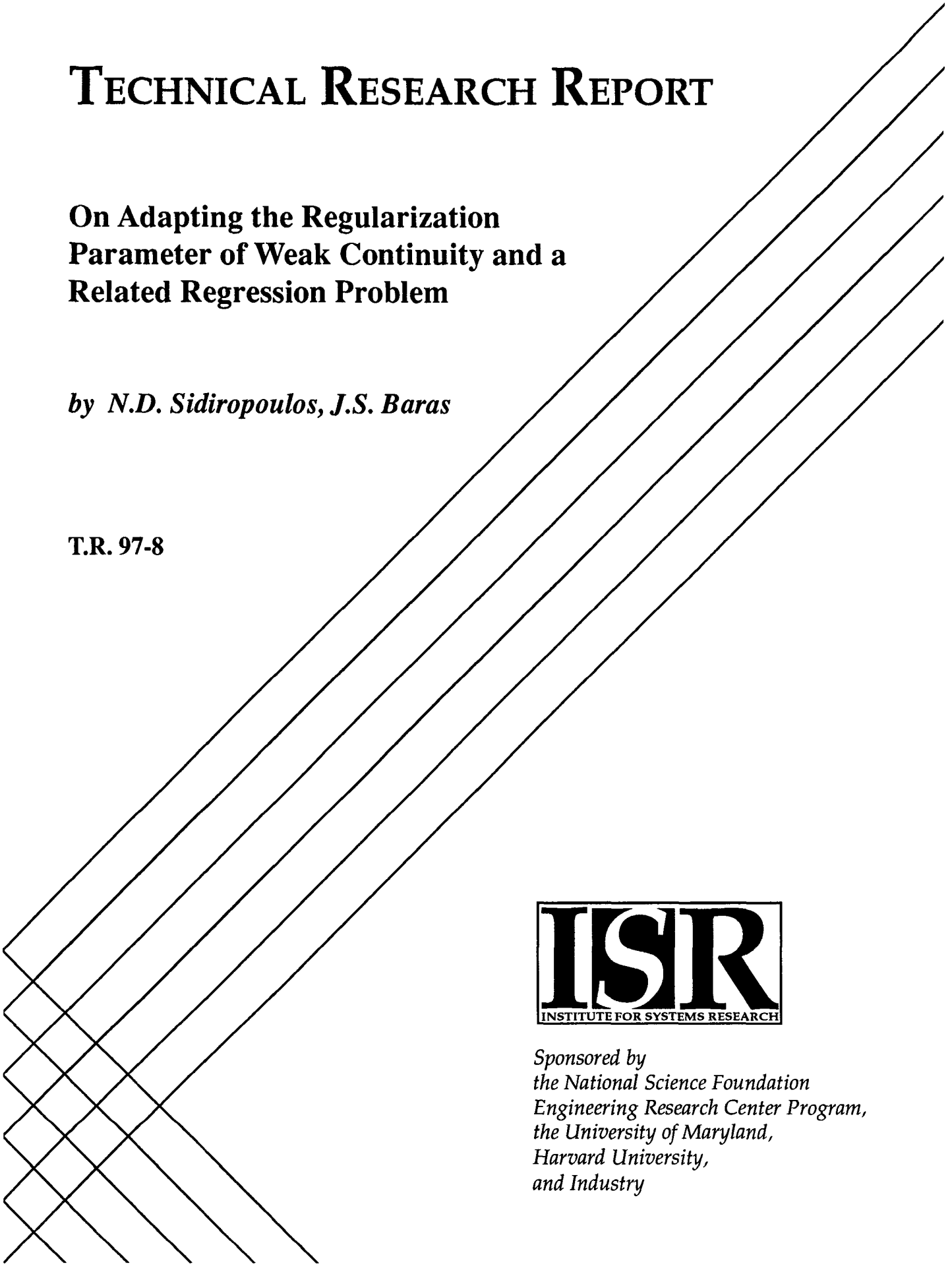
TECHNICAL RESEARCH REPORT

On Adapting the Regularization Parameter of Weak Continuity and a Related Regression Problem

by N.D. Sidiropoulos, J.S. Baras

T.R. 97-8



Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

On Adapting the Regularization Parameter of Weak Continuity and a Related Regression Problem

N.D. Sidiropoulos, J.S. Baras

Abstract

We invoke a basic lemma from the theory of so-called *penalty methods* of nonlinear programming to come up with a simple yet highly efficient block-adaptive training procedure that selects a suitable Weak Continuity (WC) regularization parameter by matching the complexity ("prior") of the resulting solution to that of a desired response. The matching is achieved using a simple binary search technique that is guaranteed to converge in very few steps; as such, it avoids many of the potential pitfalls of other iterative methods. We also consider a "hard" counterpart of WC, provide an asymptotically optimal algorithm for solving it, and discuss connections with WC.

Keywords

Nonlinear Filtering, Weak Continuity, Segmentation, Regularization

EDICS: Primary: SP 2.7.3, Secondary: SP 6.1.7

N.D. Sidiropoulos is with the Institute for Systems Research and the Dept. of Electrical Engineering, University of Maryland, College Park, MD 20742 U.S.A. He can be reached via e-mail at nikos@Glue.umd.edu

J.S. Baras is with the Institute for Systems Research and the Dept. of Electrical Engineering, University of Maryland, College Park, MD 20742 U.S.A. He can be reached via e-mail at baras@src.umd.edu

I. Introduction

Weak Continuity (WC) is an optimization that finds application in nonlinear filtering, and, in particular, segmentation and edge detection. It has been proposed and studied by Mumford-Shah [1], [2] and Blake-Zisserman [3]. Its digital counterpart, in broad terms, can be stated as follows: Given a sequence of finite extent, $\mathbf{y} = \{y(n)\}_{n=0}^{N-1} \in \mathbf{R}^N$, find a finite-alphabet sequence, $\widehat{\mathbf{x}}_{\lambda_{WC}^2} = \left\{\widehat{x}_{\lambda_{WC}^2}(n)\right\}_{n=0}^{N-1} \in \mathcal{A}^N$, that minimizes

$$\mathcal{J}_{\lambda_{WC}^2}(\mathbf{y}, \mathbf{x}) = d(\mathbf{y}, \mathbf{x}) + \lambda_{WC}^2 c(\mathbf{x})$$

where $d(\mathbf{y}, \mathbf{x}) = \sum_{n=0}^{N-1} d_n(y(n), x(n))$ is a *distortion measure* (i.e., a cost that measures fidelity to the observed data), $c(\mathbf{x}) = \sum_{n=1}^{N-1} g_n(x(n), x(n-1))$ is a *roughness-complexity measure*, and λ_{WC}^2 is a regularization parameter. From a Bayesian perspective, WC may be interpreted as *Maximum A Posteriori* (MAP) estimation of a first-order Markov signal embedded in i.i.d. noise, provided that $d_n(y(n), x(n))$ is taken to be $d(y(n)-x(n))$, where $d(\cdot)$ is proportional to minus the logarithm of the noise marginal, and $c(\mathbf{x})$ reflects minus the logarithm of the signal prior. Higher-order Markov models and/or correlated noise may be handled by an appropriate state expansion; this entails a significant increase in complexity. Throughout, we consider $\mathbf{y}$ to be given, and drop the dependence of $d()$, $\mathcal{J}_{\lambda_{WC}^2}()$ on $\mathbf{y}$. In addition, for brevity, we usually refer to $c()$ simply as *complexity*, and to $d()$ simply as *distortion*.

WC is so called because it can be thought of as using a *weak, elastic membrane* to *fit noisy data*. *Elasticity* implies that the resulting fit is, generally speaking, *smooth*; however, as a result of membrane *weakness*, the fit may occasionally be *torn* to follow a *significant* feature (e.g., edge) in the data. WC is fundamentally a trade-off between fidelity to the observed data and complexity of the solution. This interplay is governed by λ_{WC}^2. The choice of λ_{WC}^2 has a direct impact on the solution.

The problem of choosing a regularization parameter in signal and image restoration is of great enough interest to have been addressed in several special cases in the signal processing literature; see [7], the follow-up work [8], [9], and references therein, in particular [10]. Common approaches include *constrained least squares (CLS)* [11], the *chi-squared choice, generalized cross-validation, equivalent degrees of freedom*, and several possible Bayesian approaches [9].

These methods primarily address the problem of choosing the regularization parameter for a least squares formulation in which roughness is measured via a linear operator. In particular,

the objective function to be minimized incorporates a roughness-complexity measure that is the square of the Euclidean norm of a linear high-pass operator acting on $\mathbf{x} \in \mathbf{R}^N$, and the solution (and the resulting I/O map) is *explicit* and *linear* in the observation $\mathbf{y}$. In contrast, WC employs a roughness-complexity penalty that is assessed via a nonlinear operator, its solution (and the resulting I/O map) is *highly nonlinear*, and, except for trivial special cases, it cannot be written in closed form; it has to be computed *algorithmically.* Thus the above methods are not directly applicable to the problem of choosing the regularization parameter of WC, although, as we will see in section III, there exists some common ground.

In this paper, we invoke a key lemma from the theory of so-called *penalty methods* of nonlinear programming [4], [5] to come up with a simple yet highly efficient block-adaptive training procedure that selects a suitable λ_{WC}^2 by matching the complexity of the resulting solution to that of a desired response. This is close, in spirit, to traditional block-adaptive least squares (LS) [6]. Unlike LS adaptation, the procedure proposed herein is based on *partial* but *critical* side information about the desired response. As a result of the former attribute the procedure is robust; as a result of the latter, it retains sufficient flexibility to allow for proper adaptation. The matching *per se* is achieved using a simple binary search technique that is guaranteed to converge in very few steps; as such, it avoids many of the potential pitfalls of other iterative adaptation methods. Guaranteed fast convergence implies that the "filter" can occasionally be retrained without sustaining significant overhead. We call the proposed fast matching procedure *complexity matching pursuit.*

Complexity matching pursuit, when viewed from a statistical perspective, amounts to matching the *a priori* probability of the solution (as measured with respect to a given model prior) to the *a priori* probability of a reference signal. This is different from the Bayes and empirical Bayes approaches in [9], which suggest using various conditional modes for choosing a suitable regularization parameter value.

WC employs a "soft" complexity constraint, in the sense that high complexity solutions are penalized but not disqualified *a priori*; and the incurred penalty is proportional to complexity, the proportionality constant being λ_{WC}^2. A related optimization is the following, which we shall dub CLA, for Complexity-Limited Approximation:

$$\textbf{minimize} \quad d(\mathbf{x})$$

$$\textbf{subject to}: c(\mathbf{x}) \leq P, \ \mathbf{x} \in \mathcal{A}^N$$

Here, again, $d(\mathbf{y},\mathbf{x}) = \sum_{n=0}^{N-1} d_n(y(n),x(n))$, $c(\mathbf{x}) = \sum_{n=1}^{N-1} g_n(x(n),x(n-1))$, and we further assume that $c(\mathbf{x})$ is a non-negative integer $\forall \mathbf{x} \in \mathcal{A}^N$. A special instance of this latter problem dates back to the work of Bellman [12]. CLA employs a *hard* complexity constraint. We will provide an asymptotically optimal $O(|\mathcal{A}|^2 PN)$ Dynamic Programming (DP) solution to this latter problem, compare it with Bellman's solution ($O(PN^2)$) [12], discuss the relative merits of WC versus CLA, and shed light into the connection between the two. In particular, we show how WC can be used to approximately solve the CLA problem in $O(|\mathcal{A}|^2 N)$; compare with $O(|\mathcal{A}|^2 PN)$ for our exact solution, and Bellman's $O(PN^2)$ exact solution [12]. We will also verify the validity of our results by means of simulation.

The idea that one may obtain a solution to a constrained optimization problem by solving instead a suitable unconstrained problem employing an objective function that looks like

$$\mathcal{J}_\lambda(\mathbf{x}) = \mathcal{J}(\mathbf{x}) + \lambda c(\mathbf{x})$$

where $\mathcal{J}(\mathbf{x})$ is the objective function of the original constrained problem, $c(\mathbf{x})$ is a measure of conformity of $\mathbf{x}$ to the given constraint, and $\lambda > 0$, is certainly a very familiar one: this is the basic idea behind the method of Lagrange multipliers [4]. However, this latter method is not directly applicable to optimization problems involving discrete variables, like CLA [13]. In general, classical optimization techniques, like steepest descent, are not applicable to nonconvex problems [13], like WC, even if these problems only involve continuous variables; this is due to the existence of local minima [13]. We emphasize that WC solutions have to be computed *algorithmically* for each value of the regularization parameter. We use DP over "time" [14] to obtain an exact solution to WC for each such parameter value. In contrast, the method of Lagrange multipliers usually results in a parametric solution, parameterized by the regularization parameter; one then chooses this parameter to satisfy the constraint of the original constrained optimization problem. Since we have to solve WC for each value of the regularization parameter, we have to choose a suitable value for this parameter *iteratively* i.e., by trial and error. This paper proposes, among other things, a simple yet effective way to do this in an intelligent fashion.

A. Organization

The rest of this paper is organized as follows. Section II provides some background on WC and related themes. Section III describes the proposed method for adapting the WC regularization parameter. Section IV provides details of two complete adaptation suites. Section V discusses

the CLA problem and an alternative Viterbi algorithm for solving it (C-like pseudo-code can be found in the Appendix); Finally, Section VI discusses the connections between WC and CLA, while Section VII summarizes conclusions and points to further research directions.

II. Background

One of the classic problems of nonlinear filtering is that of detecting and estimating edges in noise. Among the many approaches proposed so far, a particularly noteworthy one is WC, proposed and studied by Mumford-Shah [1], [2] and Blake-Zisserman [3] (see also Morel and Solimini [15]). Weak continuity attempts to fit *piecewise*-smooth candidate "interpretations" to the observable data.

Following Blake and Zisserman [3], we present a digital version of discrete-time WC. Given a (generally real-valued) sequence of finite extent $\mathbf{y} = \{y(n)\}_{n=0}^{N-1} \in \mathbf{R}^N$, find a finite-alphabet sequence, $\widehat{\mathbf{x}} = \{\widehat{x}(n)\}_{n=0}^{N-1} \in \mathcal{A}^N$ (the "reproduction process"; in practice $\mathcal{A}$ is e.g. $\{0, 1, \cdots, 255\}$), and a sequence of boolean "edge markers", $\widehat{\mathbf{e}} = \{\widehat{e}(n)\}_{n=1}^{N-1} \in \{0,1\}^{N-1}$ (the "edge process"), so that the following cost is minimized

$$\mathcal{V}_{WC}(\mathbf{y}, \mathbf{x}, \mathbf{e}) = \sum_{n=0}^{N-1} (y(n) - x(n))^2 + \sum_{n=1}^{N-1} \Big[\lambda_{WC}^2 (x(n) - x(n-1))^2 (1 - e(n)) + \alpha e(n)\Big]$$

Here, α is a non-negative real. Intuitively, if $(x(n) - x(n-1))^2$ is too large, one has the option of declaring an "edge" in between $x(n)$ and $x(n-1)$ by choosing $e(n) = 1$, and thus paying only α, instead of $\lambda_{WC}^2 (x(n) - x(n-1))^2$. One can first minimize with respect to the edge process, then minimize the resulting functional with respect to the reproduction process. Since the first sum in the combined cost does not depend on the edge process, it is easy to see [3] that the optimization above is equivalent to minimizing

$$\mathcal{V}'_{WC}(\mathbf{y}, \mathbf{x}) = \sum_{n=0}^{N-1} (y(n) - x(n))^2 + \sum_{n=1}^{N-1} h_{\alpha, \lambda_{WC}}(x(n) - x(n-1))$$

by appropriate choice of reproduction process, $\mathbf{x}$, where $h_{\alpha,\lambda_{WC}} : \mathbf{Z} \mapsto \mathbf{R}$ is defined as

$$h_{\alpha,\lambda_{WC}}(t) = \begin{cases} \lambda_{WC}^2 t^2 & , \; t^2 < \frac{\alpha}{\lambda_{WC}^2} \\ \alpha & , \; otherwise \end{cases}$$

The associated optimal edge process can be implicitly inferred, once the optimal reproduction process is determined, by level tests on the first order residuals, $\widehat{x}(n) - \widehat{x}(n-1)$, of the optimal reproduction process.

A. Solving WC

There exist essentially two ways to go about solving WC: DP [16], [17], [18], [19], and the so-called *Graduated Non Convexity (GNC)* algorithm [3]. The GNC is suitable for optimization over $\widehat{\mathbf{x}} \in \mathbf{R}^N$, i.e., the continuous-valued case, and it does not lend itself to discrete-valued problems, i.e., $\widehat{\mathbf{x}} \in \mathcal{A}^N$ [13]. DP is exact, i.e., it provides a true minimizer; GNC has been proven to do so for a large class of inputs [3], but not for an arbitrary input. The drawback of DP is that it becomes computationally very demanding in higher dimensions. The GNC, by comparison, carries over quite effortlessly in higher dimensions. The GNC is a special case of *Mean Field Annealing* [20].

Let us briefly review the basic idea behind DP.

A.1 Dynamic Programming: The Viterbi Algorithm

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:

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

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.

Simplistic as it may sound, this is one of the most powerful and pervasive optimization techniques [35], and it finds application in many diverse areas, including, but not limited to:

- Optimum decoding of convolutional codes [36], [37], [38].
- Speech and character recognition [39].
- State estimation in Hidden Markov Models (HMM's) [39].
- Optimal Control [19].
- Game theory [19].

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 [40], [41], [23].

A.2 Choice of DP algorithm

There are two DP algorithms for WC, one that works by DP over "time", and *requires* $\mathbf{x}$ to be quantized [14]; and another that works by DP over "edges" [13], and works for either continuous or discrete-valued $\mathbf{x}$, i.e., either $\widehat{\mathbf{x}} \in \mathbf{R}^N$, or $\widehat{\mathbf{x}} \in \mathcal{A}^N$. The latter is much slower than the former for moderate $|\mathcal{A}|$. Here we consider the discrete-valued problem, so we opt for the former; throughout, we use forward DP over "time" (the VA) to solve WC in $O(|\mathcal{A}|^2 N)$.

As mentioned earlier, WC can be interpreted from a Bayesian estimation viewpoint, and is closely related to MAP inference for Markov models and associated *annealing*-type algorithms [24], [20], [25], [26], [27], [28], [29], [30], [31], [32], [33]. In this context, the complexity measure reflects the signal prior, and, therefore, matching complexity may be interpreted as matching the *a priori* probability of the solution (as measured with respect to a given model prior) to the *a priori* probability of a reference signal.

III. A method for choosing the WC regularization parameter

A. Key Lemma

The following result appears in the theory of so-called *penalty methods* in *nonlinear programming* [4], [5].

Lemma 1: Consider:

$$WC: \quad \widehat{\mathbf{x}}_{\lambda_{WC}^2} \leftarrow min J_{\lambda_{WC}^2}(\mathbf{x})$$

where

$$J_{\lambda_{WC}^2}(\mathbf{x}) = d(\mathbf{x}) + \lambda_{WC}^2 c(\mathbf{x})$$

Then:

$$d(\widehat{\mathbf{x}}_{\lambda^2_{WC}}) \quad \uparrow \quad \lambda^2_{WC}$$

$$c(\widehat{\mathbf{x}}_{\lambda^2_{WC}}) \quad \downarrow \quad \lambda^2_{WC}$$

where $\uparrow$ stands for *non-decreasing in*, and $\downarrow$ stands for *non-increasing in.*

Proof: This works for *any* $d(\cdot), c(\cdot)$, and $\mathbf{x}$ in some *arbitrary* space. In the theory of penalty methods it is assumed that $d(\cdot), c(\cdot)$ are continuous functions in continuous variables, and $c(\mathbf{x}) \geq 0$, $\forall \mathbf{x}$, yet these are really needed for reasons other than the proof of this Lemma [4, pp. 279]. ■

The basic idea of adaptation is depicted in figure 1. This is a block-adaptive training procedure, similar in spirit to block-adaptive LS [6]. In training mode, we assume that we have access to a block of training data and a desired response block, which are representative of the operational environment which will be encountered during actual runtime. The purpose is to adapt the λ^2_{WC} WC parameter to match the output of WC to the desired response. What we propose here is to select a suitable λ^2_{WC} by matching the complexity of the resulting solution (WC output) to that of the desired response, instead of matching the output *per se* to the desired response in a LS sense. This is motivated by two observations.

- WC is fundamentally a trade-off between fidelity to the observed data and complexity of the solution; therefore the complexity of the desired solution is a critical and highly informative attribute. As we will see, matching complexity affords enough flexibility to allow for meaningful and effective adaptation.
- Matching the WC output to the desired response in a LS sense seems intractable, due to (i) the nonlinearity of the WC I/O map, and (ii) the lack of an analytical closed-form solution (in contrast, in the LTI FIR case the I/O map is linear and of very simple closed form). On the other hand, matching the complexity of the solution (WC output) to that of the desired response is made particularly easy by Lemma 1.

Had we known the complexity versus λ^2_{WC} curve, we could have selected an appropriate λ^2_{WC} by inspection. We do not, hence we have to search for a suitable λ^2_{WC} in an intelligent fashion. Lemma 1 tells us that complexity is nonincreasing in λ^2_{WC} (a typical plot is presented in Figure 8); thus, even though we do not know the specifics of the complexity versus λ^2_{WC} curve, we may search for a suitable λ^2_{WC} using a simple *binary search* technique:

- Select a candidate range for λ^2_{WC}: since for $\lambda^2_{WC} > min_{\mathbf{x}_0} d(\mathbf{x}_0)$, where $c(\mathbf{x}_0) = 0$, further increase in λ^2_{WC} does not affect the solution, we may choose the initial range: $\lambda^2_{WC} \in [0, min_{\mathbf{x}_0} d(\mathbf{x}_0)]$, where $c(\mathbf{x}_0) = 0$. Let l, r denote the limits of the current search range. Thus, in the beginning, we set $l = 0$, $r = min_{\mathbf{x}_0} d(\mathbf{x}_0)$, where $c(\mathbf{x}_0) = 0$.
- Run WC with $\lambda^2_{WC} = m = \frac{l+r}{2}$.
- If the complexity of the resulting solution is more than that of the desired response, then set: $l = m$; else if it is less than that of the desired response, then set: $r = m$; else if the same exit;
- Repeat the last two steps until some prespecified accuracy in the localization of λ^2_{WC} is reached (i.e., $|l - r| <$ prespecified accuracy), or the target complexity is attained.

Due to binary search, the effective search range is at least halved in each iteration; convergence is therefore logarithmic in the effective range of λ^2_{WC}. This means that less than 10 iterations are usually enough.

In practice, it pays to use an integer λ^2_{WC} for it allows for integer trellis arithmetic. In addition, complexity as a function of λ^2_{WC} typically exhibits a piecewise-constant behavior. As a result, a complexity-matching value of λ^2_{WC} may be highly non-unique. In view of these, it is beneficial to quantize λ^2_{WC} beforehand (this may require appropriate scaling), and execute a discrete binary search. In this case, the statement "until some prespecified accuracy is reached" should be replaced by "until $|r - l| \leq q$", where q is the λ^2_{WC}-quantization step.

Each run of WC takes $O(|\mathcal{A}|^2 N)$. If we quantize λ^2_{WC} to Λ levels, then the entire adaptation takes $O(log(\Lambda)|\mathcal{A}|^2 N)$. Since we are matching complexity in an intelligent way (by binary search), it is appropriate to call the procedure a *complexity matching pursuit (CMP).*

Complexity matching pursuit is close, in spirit, to a classic polynomial root-finding technique, known as *bisection* [34]. The difference here is that, at each step, we implement WC via DP, instead of simply evaluating a polynomial. It is also close in spirit to CLS [11], which is an iterative technique that may be used to select the regularization parameter in such a way that the solution to the resulting optimization problem attains a prespecified level of either distortion or roughness. CLS also builds on monotonicity, and uses a Newton-Raphson-like iteration. The differences between CLS and CMP are summarized below.

- CLS has been developed for choosing the regularization parameter for a least squares problem in which roughness is measured via a linear operator. CMP addresses the same problem for

WC, in which roughness is measured via a nonlinear operator. In general, WC need not employ the square of the Euclidean norm.

- CLS matches distortion, whereas CMP matches complexity. CLS may oversmooth the data (e.g., cf. [7] and references therein). In the context of segmentation, CMP matches *number of segments*, a much more informative attribute, and as a result the matching is more effective.
- In CLS there is always a unique value of the regularization parameter that exactly satisfies the constraint. In WC-CMP this is typically not the case.
- CLS uses a Newton-Raphson-like iteration, which is not suitable for integer problems due to the possibility of oscillations. CMP uses a binary search that is guaranteed to converge to an optimal discrete value of the regularization parameter. Recall that a *discrete* value is sought for the sake of reduced runtime complexity, as it translates to integer trellis arithmetic. This benefit of binary search versus Newton-Raphson is only exhibited in the case of discrete-parameter problems.
- For each interim value of the regularization parameter, the CLS signal estimate is *explicit*, and *linear* in the observation. At each step, the corresponding WC-CMP estimate has to be computed *algorithmically*, using Dynamic Programming.

Obviously, a similar adaptation can be proposed on the basis of distortion, which is the other aspect of WC. This gives rise to a *distortion matching pursuit*, which, in general, leads to a different choice of λ_{WC}^2 than complexity matching pursuit. However, just like complexity, distortion as a function of λ_{WC}^2 exhibits a piecewise-constant behavior. With twice the original matching pursuit effort, one may identify candidate *ranges* of λ_{WC}^2 and check for intersection. We would like to point out that distortion matching pursuit appears to be less powerful than complexity matching pursuit. This can be attributed to the fact that complexity carries more prior information about the desired response than distortion does. We thus focus on complexity matching pursuit.

IV. Simulation

Let us now present two complete WC adaptation experiments. These experiments use complexity matching pursuit; here, due to the special choice of $g_n(x(n), x(n-1))$ (see below) complexity is tantamount to number of edges.

The first experiment is concerned with the estimation of a synthetic edge signal embedded in noise, using WC. Figure 3 depicts the input sequence. This particular input has been generated

by adding i.i.d. noise on the data depicted in Figure 2. This latter data is also overlaid on subsequent plots using a dashed line style. The target complexity of the desired response is 9.

For this example, we take $d_n(y(n), x(n)) = |y(n) - x(n)|$, $\forall n \in \{0, 1, \cdots, N-1\}$, $g_n(x(n), x(n-1)) = 1 - \delta(x(n) - x(n-1))$, $\forall n \in \{1, \cdots, N-1\}$, $\mathcal{A} = \{0, \cdots, 99\}$, and $N = 512$. Four iterations of WC adaptation on the basis of the desired response of Figure 2, and the input data of Figure 3, are depicted in Figures 4, 5, 6, and 7, respectively. The plots are self-explanatory. The initial choice of λ^2_{WC} is really irrelevant; it is corrected in one iteration step.

Figures 8, 9, depict plots of $c(\widehat{\mathbf{x}}_{\lambda^2_{WC}})$ versus λ^2_{WC}, and $d(\widehat{\mathbf{x}}_{\lambda^2_{WC}})$ versus λ^2_{WC}, respectively, for the input data of Figure 3. These have been computed essentially by *brute-force*. They are not needed for adaptation, and they are presented here to build the reader's understanding and intuition.

The second experiment is concerned with a segmentation problem. Figure 10 depicts left ventricular pressure data (dog), from the Signal Processing Information Base at www.spib.rice.edu. A noisy version of this data (corrupted by AWGN) is depicted in Figure 11. The noise-free data is quasi-periodic, and the goal is to segment the pulses. By periodicity, the target complexity is 5.

For this example, we take $d_n(y(n), x(n)) = |y(n) - x(n)|^2$ (consistent with the fact that the noise is Gaussian), and $g_n(x(n), x(n-1)) = 1 - \delta(x(n) - x(n-1))$. We use a total of $|\mathcal{A}| = 200$ reproduction levels. Two iterations of WC adaptation are depicted in Figures 12, and 13. For this data the pursuit ends in just two steps.

V. CLA

Let us now return to the CLA approximation problem:

$$\textbf{minimize} \quad d(\mathbf{x})$$

$$\textbf{subject to} : c(\mathbf{x}) \leq P, \ \mathbf{x} \in \mathcal{A}^N$$

Recall that $d(\mathbf{y}, \mathbf{x}) = \sum_{n=0}^{N-1} d_n(y(n), x(n))$, $c(\mathbf{x}) = \sum_{n=1}^{N-1} g_n(x(n), x(n-1))$, and we further assume that $c(\mathbf{x})$ is a non-negative integer $\forall \mathbf{x} \in \mathcal{A}^N$. This problem can be solved using Dynamic Programming. A particular instance of this latter problem, namely, for $c(\mathbf{x}) = \sum_{n=1}^{N-1} g_n(x(n), x(n-1)) = \sum_{n=1}^{N-1} [1 - \delta(x(n) - x(n-1))]$ (i.e., complexity is measured by number of edges), and $d(\mathbf{x}) = \sum_{n=0}^{N-1} |y(n) - x(n)|^2$, is a special case of the problem of fitting curves by line segments subject to an upper bound on the number of segments that can be used, and under a quadratic

fidelity criterion. This has been solved in 1961 by Bellman [12]. Bellman's solution employs DP *over number of segments*, i.e., his recursion is over the number of segments; the resulting DP algorithm is $O(PN^2)$ [12]. An alternative way to approach the CLA problem is by DP *over "time"*, i.e., over the N variable. The motivation is that DP is always linear in the recursion variable, and this may be beneficial for large N.

A. CLA solution by DP over "time"

The CLA problem involves a *global* rather than *local* constraint. As such, it doesn't naturally fall along the lines of [40], [41], [23]. Nevertheless, one can readily see that the pair (value(n), complexity-accrued-sofar(n)) is a *state* for the CLA problem, which allows one to recast CLA in the form of minimizing a cost of the type that appears in Equation 1, and, therefore, set up a suitable Viterbi-type *trellis* to solve it. A fairly detailed yet compact C-like Viterbi-type algorithm to solve CLA (for the special case of measuring complexity by number of segments, and $d(\mathbf{x}) = \sum_{n=0}^{N-1} |y(n) - x(n)|$; it easily generalizes to other choices) can be found in the Appendix. It is also straightforward to see that the resulting complexity is $O(|\mathcal{A}|^2 PN)$. For fixed $\mathcal{A}, P$ this is asymptotically optimal in N; depending on the particular value of $|\mathcal{A}|$, it may be worse than Bellman's $O(PN^2)$ original solution for small to moderate N, but it clearly beats it for larger N.

VI. Connections between WC and CLA

We now turn our attention to the following problem:

$$\textbf{minimize} \quad d(\mathbf{x}) + \lambda^2 T_P(c(\mathbf{x}))$$

over $\mathbf{x} \in \mathcal{A}^N$, where

$$T_P(l) = \begin{cases} 0 & , l \leq P \\ 1 & , l > P \end{cases}$$

It is easy to see that if $d(\mathbf{x}) \geq 0, \ \forall \mathbf{x}$, $c(\mathbf{x}) \geq 0, \ \forall \mathbf{x}$, and $\lambda^2 > min_{\mathbf{x}_0} d(\mathbf{x}_0)$, where $c(\mathbf{x}_0) = 0$, then the minimization above is, in fact, equivalent to CLA. It also looks a lot like WC, which is reproduced here for convenience:

$$\textbf{minimize} \quad d(\mathbf{x}) + \lambda_{WC}^2 c(\mathbf{x})$$

over $\mathbf{x} \in \mathcal{A}^N$. The difference between the two lies in the fact that the latter employs a penalty that is proportional to complexity, whereas the former employs a penalty that is a nonlinear

function of complexity. As a result, the former (which, by suitable choice of λ^2, can be made equivalent to CLA) does not directly admit a Viterbi solution; rather it requires a state expansion, just like CLA, and this leads to complexity $O(|\mathcal{A}|^2 PN)$, as for CLA. On the other hand, WC is directly amenable to Viterbi solution, and the resulting complexity is $O(|\mathcal{A}|^2 N)$. This, along with the fact that the WC regularization parameter can be automatically tuned to match complexity to a prespecified number, suggests that one may be able to train WC to provide at least an approximate solution to the CLA problem in time $O(|\mathcal{A}|^2 N)$ (compare with $O(|\mathcal{A}|^2 PN)$ for our exact solution of CLA, and Bellman's $O(PN^2)$ exact solution [12]). This is indeed the case. Consider Figure 14. It depicts the solution of CLA for $P = 9$, complexity measured by number of edges, and $d(\mathbf{x}) = \sum_{n=0}^{N-1} |y(n) - x(n)|$. The result is actually identical to that of Figure 7. This can be explained as follows: once the WC parameter is tuned to match the complexity of WC output to P, the hard upper bound on complexity in the statement of CLA, and assuming that the CLA problem is non-degenerate (in the sense that the optimum is not achieved using fewer than P segments), then WC picks the best among all solutions of complexity P *by simply minimizing distortion*, which is precisely what the CLA does. This assumes, again, that: (i) the WC parameter *can* be selected to match the complexity of WC output to P (potential problem: discontinuities in the complexity vs. λ_{WC}^2 curve), (ii) the CLA problem is non-degenerate for the given P, and (iii) the training data is indeed representative of the actual operational environment. In practice, these considerations imply that, in general, WC (properly tuned) only provides an *approximate* solution to the CLA problem; this is, however, often good enough.

A final note is in order. It is quite obvious that CLA can be designed to provide a solution of a given complexity (providing results comparable to adaptive WC) without any adaptation effort. The drawback is, of course, increased runtime computational complexity with respect to WC. Thus, adaptive WC can be viewed as successfully trading on-line computational complexity for a modest off-line training effort.

VII. Summary

We have proposed a simple yet highly efficient block-adaptive training procedure that selects a suitable Weak Continuity (WC) regularization parameter by matching the complexity of the resulting solution to that of a "desired response". The matching is achieved using a simple binary search technique that is guaranteed to converge in very few steps. This technique may be useful in the context of other related nonlinear regularization problems.

We have also considered CLA, a "hard" counterpart of WC, provided an asymptotically optimal algorithm for solving it, and discussed connections with WC. In particular, we discussed the potential of WC to provide an approximate but computationally cheaper solution to the CLA problem.

It would be interesting to investigate the potential of matching pursuit in other related regularization problems. The next logical step would be to consider the adaptation of $d(\mathbf{y}, \mathbf{x}) = \sum_{n=0}^{N-1} d_n(y(n), x(n))$ and $c(\mathbf{x}) = \sum_{n=1}^{N-1} g_n(x(n), x(n-1))$. This is a difficult problem that involves many more degrees of freedom. From a Bayesian viewpoint, these should reflect the noise characteristics, and signal prior, respectively; from an MDL perspective, the second should be commensurate to model coding complexity. Other perspectives may also be appropriate. Work in this direction is currently underway.

REFERENCES

[1] D. Mumford and J. Shah, "Boundary detection by minimizing functionals", in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Francisco, 1985.

[2] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems", *Communications on Pure and Applied Math.*, vol. 42, pp. 577–685, 1989.

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

[4] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, Massachusetts, 1973.

[5] D. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1995.

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

[7] Nikolas P. Galatsanos and Aggelos K. Katsaggelos, "Methods for choosing the regularization parameter and estimating the noise variance in image restoration and their relation", *IEEE Trans. on Image Processing*, vol. 1, no. 3, pp. 322–336, July 1992.

[8] M. G. Kang and Aggelos K. Katsaggelos, "General choice of the regularization functional in regularized image restoration", *IEEE Trans. on Image Processing*, vol. 4, no. 5, pp. 594–602, May 1995.

[9] G. Archer and D. M. Titterington, "On some Bayesian/Regularization methods for image restoration", *IEEE Trans. on Image Processing*, vol. 4, no. 7, pp. 989–995, July 1995.

[10] Grace Wahba, *Spline Models for Observational Data*, vol. 59 of *CBNS*, SIAM, 1990.

[11] B. R. Hunt, "The application of constrained least squares estimation to image restoration by digital computer", *IEEE Trans. on Computers*, vol. C-22, no. 9, pp. 805–812, September 1973.

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

[13] 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.

[14] A.V. Papoulias, "Curve Segmentations Using Weak Continuity Constraints", M.Sc. thesis, Univ. of Edinburgh, 1985.

[15] Jean-Michel Morel and Sergio Solimini, *Variational Methods in Image Segmentation*, Birkhauser, Boston-Basel-Berlin, 1994.

[16] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, N.J., 1957.

[17] R. Bellman and S. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J., 1962.

[18] S. Dreyfus and A. Law, *The Art and Theory of Dynamic Programming*, Academic, New York, NY, 1977.

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

[20] G.L. Bibro, W.E. Snyder, S.J. Garnier, and J.W. Gault, "Mean field annealing: a formalism for constructing GNC-like algorithms", *IEEE Trans. Neural Networks*, vol. 3, no. 1, pp. 131–138, Jan. 1992.

[21] Y. Leclerc, "Constructing Simple Stable Descriptions for Image Partitioning", *Int. J. Computer Vision*, vol. 3, no. 1, pp. 73–102, 1989.

[22] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific, 1989.

[23] 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.

[24] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images", *IEEE Trans. PAMI*, vol. PAMI-6, no. 6, pp. 721–741, November 1984.

[25] D. Geman, S. Geman, C. Graffigne, and P. Dong, "Boundary detection by constrained optimization", *IEEE Trans. PAMI*, vol. PAMI-12, no. 7, pp. 609–627, 1990.

[26] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization", *IEEE Trans. Image Processing*, vol. 4, no. 7, pp. 932–946, July 1995.

[27] D. Geman and G. Reynolds, "Constrained restoration and the recovery of discontinuities", *IEEE Trans. PAMI*, vol. PAMI-14, no. 3, pp. 367–384, Mar. 1992.

[28] D. Geiger and F. Girosi, "Parallel and deterministic algorithms from MRF's: surface reconstruction", *IEEE Trans. PAMI*, vol. PAMI-13, no. 5, pp. 401–412, May 1991.

[29] M. Miller, B. Roysam, K. Smith, and J. O'Sullivan, "Representing and computing regular languages on massively parallel networks", *IEEE Trans. Neural Networks*, vol. 2, no. 1, pp. 56–72, Jan. 1991.

[30] P.M. Djuric, "A MAP solution to the off-line segmentation of signals", in *Proc. IEEE ICASSP'94*, Adelaide, 1994, vol. IV, pp. 505–508.

[31] M.A.T. Figueiredo and J.M.N. Leitao, "Adaptive discontinuity location in image restoration", in *Proc. IEEE ICIP'94*, Austin, Texas, 1994, vol. I, pp. 665–669.

[32] S. Geman, D.E. McClure, and D. Geman, "A nonlinear filter for film restoration and other problems in image processing", *Computer Vision, Graphics and Image Processing: Graphical Models and Image Processing*, vol. 54, no. 4, pp. 281–289, July 1992.

[33] C. Bouman and K. Sauer, "A generalized Gaussian image model for edge-preserving MAP estimation", *IEEE Trans. on Image Proc.*, vol. 2, no. 3, pp. 296–310, July 1993.

[34] C.F. Gerald and P.O. Wheatley, *Applied Numerical Analysis*, Addison-Wesley, 1990.

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

[36] 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.

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

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

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

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

[41] N.D. Sidiropoulos, "Fast Digital Locally Monotonic Regression", *IEEE Trans. Signal Processing*, 1997, To appear.

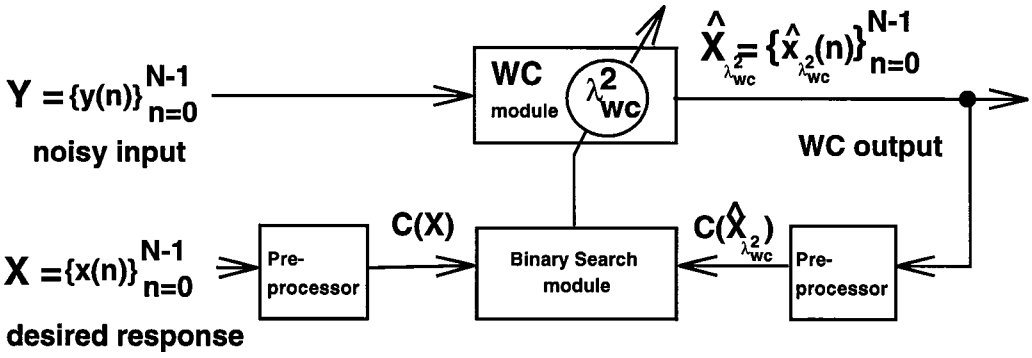


Fig. 1. Block-Adaptive WC schematic

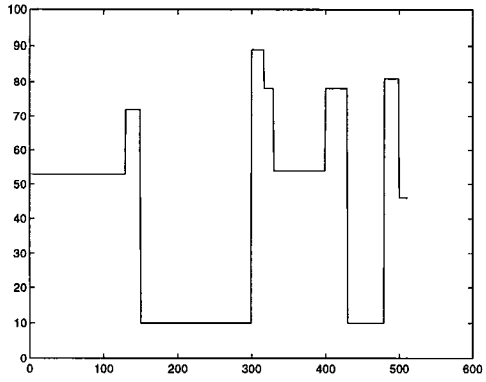


Fig. 2. Synthetic edge data, $\mathbf{x}$, $c(\mathbf{x}) = 9$

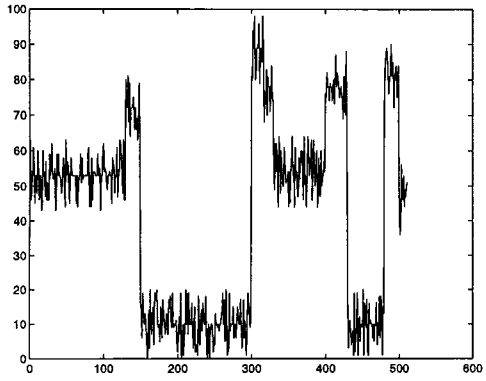


Fig. 3. Input sequence, $\{y(n)\}_{n=0}^{511}$

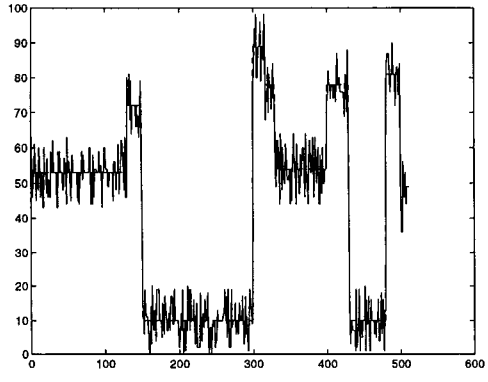


Fig. 4. WC output, ITERATION = 1, $\lambda_{WC}^2 = 2$, $c(\widehat{\mathbf{x}}_{\lambda_{WC}^2}) = 355$.

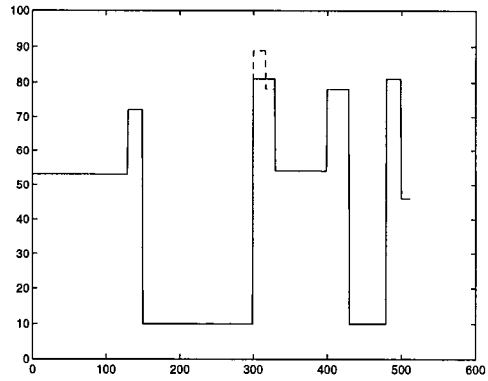


Fig. 5. WC output, ITERATION = 2, $\lambda_{WC}^2 = 151$, $c(\widehat{\mathbf{x}}_{\lambda_{WC}^2}) = 8$.

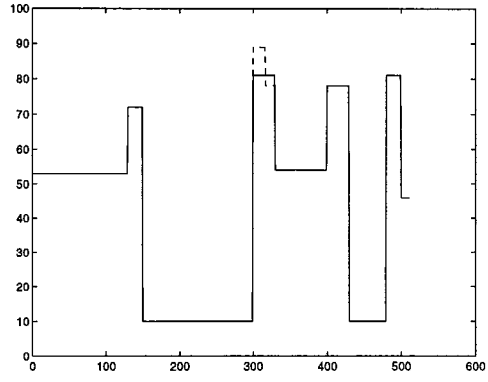


Fig. 6. WC output, ITERATION = 3, $\lambda_{WC}^2 = 76$, $c(\widehat{\mathbf{x}}_{\lambda_{WC}^2}) = 8$.

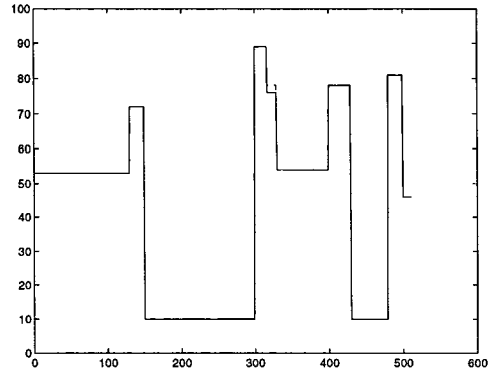


Fig. 7. WC output, ITERATION = 4, $\lambda_{WC}^2 = 39$, $c(\widehat{\mathbf{x}}_{\lambda_{WC}^2}) = 9$.

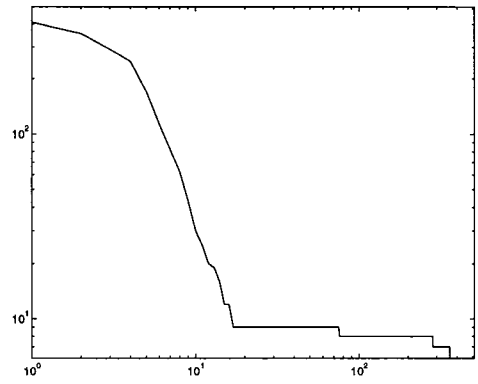


Fig. 8. Log-log plot of $c(\widehat{\mathbf{x}}_{\lambda^2_{WC}})$ versus λ^2_{WC}.

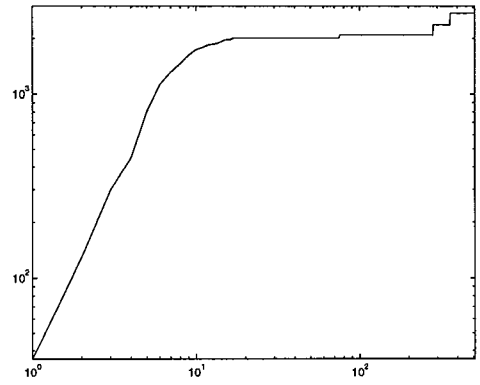


Fig. 9. Log-log plot of $d(\widehat{\mathbf{x}}_{\lambda^2_{WC}})$ versus λ^2_{WC}.

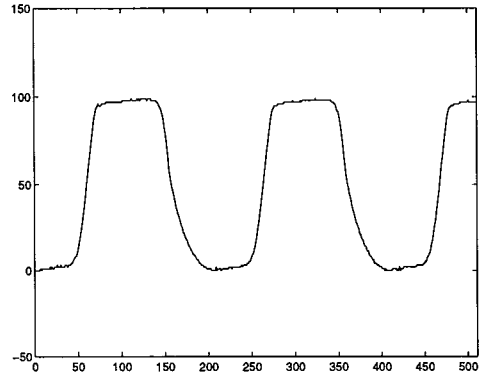


Fig. 10. Plot of left ventricular pressure (dog) (taken from SPIB)

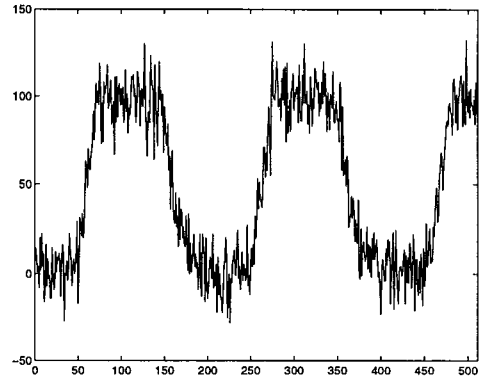


Fig. 11. Plot of left ventricular pressure (dog) corrupted by AWGN

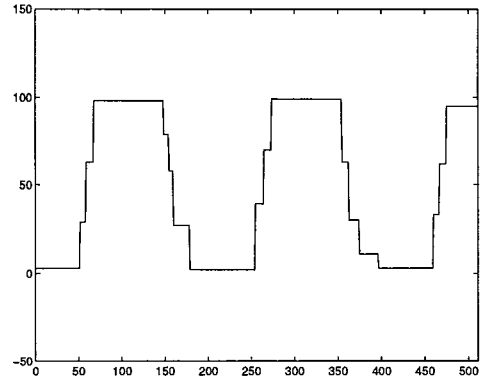


Fig. 12. WC segmentation, ITERATION = 1, $\lambda^2_{WC} = 1000$, $c(\widehat{\mathbf{x}}_{\lambda^2_{WC}}) = 17$.

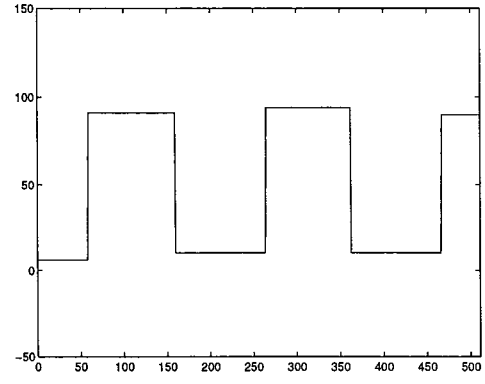


Fig. 13. WC segmentation, ITERATION = 2, $\lambda^2_{WC} = 38000$, $c(\widehat{\mathbf{x}}_{\lambda^2_{WC}}) = 5$.

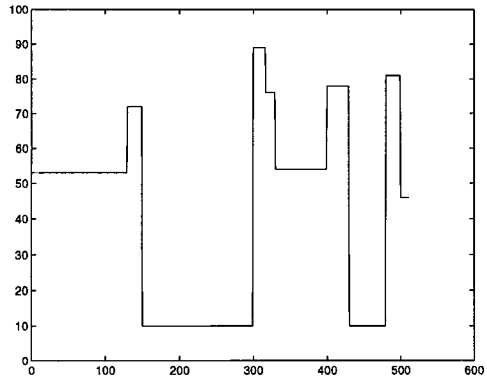


Fig. 14. CLA: Here, the input is that of Figure 3, and $P = 9$.

VIII. Appendix - CLA - VA (DP) Code

```
Uses:
typedef struct _state {
  int cumcost;              /* cost of reaching this state       */
                            /* using the best path to it         */
  struct _state *prevstate; /* pointer to best previous state    */
                            /* NULL if unreachable               */
  } state;

static state trellis[R][P][N];    /* Viterbi Trellis             */
                                  /* trellis[value][sofar][n]:   */
                                  /* state (value,sofar) at      */
                                  /* time n                      */

  Init all states in the trellis to UNREACHABLE;

  Init for n = 0 (first trellis stage); /* certain states       */
                                        /* reachable            */

  /* main loop                                                  */

  for (n = 1; n < N; n++) /* do sequentially for all n          */
  {
    for (sofar = 1; sofar <= P; sofar++) /* for all             */
    {
      for (v = 0; v < R; v++)           /* states at time n     */
      {
        /* look back @ all states @ time n-1 that are           */
        /* reachable and may lead to the given state @ time n   */
        if (sofar > 1)
        {
```

```
        /* then these states include:                              */
        for (pv = 0; pv < R; pv++)
        {
          if ((pv != v) && (trellis[pv][sofar - 1][n-1].prevstate != NULL))
          {
            cost = trellis[pv][sofar - 1][n-1].cumcost +
                    absolute_value((v-y[n]));
            if (cost < trellis[v][sofar][n].cumcost)
            {
              trellis[v][sofar][n].cumcost = cost;
              trellis[v][sofar][n].prevstate = &trellis[pv][sofar - 1][n-1];
            }
          }
        }
      }
      /* in any case, even if sofar = 1, (v,sofar) @ time n-1, */
      /* if reachable, is a valid candidate:                    */
      if (trellis[v][sofar][n-1].prevstate != NULL)
      {
        cost = trellis[v][sofar][n-1].cumcost +
                absolute_value((v-y[n]));
        if (cost < trellis[v][sofar][n].cumcost)
        {
          trellis[v][sofar][n].cumcost = cost;
          trellis[v][sofar][n].prevstate = &trellis[v][sofar][n-1];
        }
      }
    }
  }
}
```