Received September 5, 2021, accepted November 10, 2021, date of publication November 17, 2021,
date of current version November 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3128902

HOPS: A Fast Algorithm for Segmenting Piecewise Polynomials of Arbitrary Orders

JUNBO DUAN[1], QING WANG[2], AND YU-PING WANG[3,4], (Senior Member, IEEE)
[1]Key Laboratory of Biomedical Information Engineering of Ministry of Education, Department of Biomedical Engineering, School of Life Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China
[2]School of Electronic Engineering, Xidian University, Xi'an 710071, China
[3]Department of Biomedical Engineering, Lindy Boggs Center, Tulane University, New Orleans, LA 70118, USA
[4]Department of Biostatistics and Bioinformatics, Lindy Boggs Center, Tulane University, New Orleans, LA 70118, USA

Corresponding author: Junbo Duan (junbo.duan@mail.xjtu.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61771381.

ABSTRACT The segmentation of piecewise polynomial signals arises in a variety of scientific and engineering fields. When a signal is modeled as a piecewise polynomial, the key then becomes the detection of breakpoints followed by curve fitting and parameter estimation. This paper proposes HOPS, a fast High-Order Polynomial Segmenter, which is based on ℓ_0-penalized least-square regression. While the least-squares regression ensures fitting fidelity, the ℓ_0 penalty takes the number of breakpoints into account. We show that dynamic programming can be applied to find the optimal solution to this problem and that a pruning strategy and matrix factorization can be utilized to accelerate the execution speed. Finally, we provide some illustrative examples, and compare the proposed method with state-of-the-art alternatives.

INDEX TERMS Piecewise polynomials, breakpoint detection, segmentation, curve fitting, sparse modeling.

I. INTRODUCTION

Many real-world signals can be modeled as piecewise polynomials. For example, in bioinformatics, read depth signals from next generation sequencing (NGS), and $\log_2$ ratio signal from comparative genomic hybridization (CGH) are approximated as piecewise constants (polynomials of zero order) for genomic copy number variation (CNV) detection [1], [2]. In geoscience, the global temperature is modeled as a piecewise linear (polynomial of order one) to study global warming [3]. In nanotechnology, the force curve of an unfolding protein measured during the retraction phase of atomic force microscopy (AFM) can be modeled as a set of worm-like chains (WLC) or freely jointed chains (FJC) [4], which can be approximated as piecewise parabolas (polynomials of order two).

The analysis of piecewise polynomials consists of two main problems: segmentation and fitting. Most applications focus on the fitting problem. For the aforementioned AFM force curve, each piece is fitted with a WLC model; then, the contour length can be estimated, as this is an important property of proteins. It is obvious that the quality of curve fitting is highly dependent on segmentation, and a model that can combine segmentation with fitting is desirable.

Several signal processing tools can be applied, and we classify them into two categories: optimal and nonoptimal methods.

A. NONOPTIMAL METHODS

Spline [5], [6] is an elegant method for fitting signals with piecewise polynomials, however, knots must be known in advance. Therefore, most spline-based methods perform segmentation before fitting. Additionally, spline enforces *consistency regarding the joint point* [7] constraints, and piecewise-constant signals are excluded under this framework.

Circular binary segmentation (CBS) [8] employs statistical testing to detect breakpoints in a binary search manner, and this is indeed a greedy method with $N \log(N)$ computational complexity.

The change points (CP) model [3] is a parametric model used to fit piecewise linear signals, and it is solved by utilizing a Markov chain Monte-Carlo (MCMC) simulator. Since MCMC is involved, asymptotic optimality is achieved.

In [9]–[11], the authors proposed an ℓ_{21}-norm penalized least-squares cost function to smooth piecewise polynomials.

The associate editor coordinating the review of this manuscript and approving it for publication was Lorenzo Mucchi.

The ℓ_1 trend filter [7] is a method using the famous total variation (TV) [12] and ℓ_1-norm based approaches, but it is confined to piecewise linear signals. In [13], the authors extended the ℓ_1 trend filter to high-order polynomials.

Even though the solution of an ℓ_1-norm penalized or constrained least-squares objective is guaranteed by the least absolute shrinkage and selection operator (LASSO) [14] and least angle regression (LARS) [15], these methods are nonoptimal since ℓ_1-norm is a relaxation of the ℓ_0-norm for the purpose of avoiding an NP-hard problem. Some conditions (*e.g.,* the restricted isometry property (RIP), mutual coherence condition (MCC), strictly diagonally dominant condition (SDDC), *etc.* [16], [17]) are needed to guarantee that the solution supports of the ℓ_0-norm and ℓ_1-norm are consistent. In addition, the ℓ_1-norm penalizes the amplitudes of nonzero entries, yielding biased estimates [18].

The Hodrick–Prescott filtering technique [19] also falls in the same category; it uses the ℓ_2-norms of the TV of the second-order derivatives to smooth piecewise linear signals. The finite-dimensional piecewise continuous (FPC) signals [20] was proposed, and discontinuities were detected by exploiting sparsity hidden in a tight-dimensional representation space.

B. OPTIMAL METHODS

By regressing with the ℓ_0-norm, optimal segmentation can be expected. The authors in [21] proposed a brute-force search method for optimal solutions, but the computational complexity was extremely high for signals of large sizes. The researchers in [22] employed dynamic programming (DP) [23] to search a given segmentation pattern. In [24], partition regression was proposed, and the computational complexity and optimality of DP were also discussed comprehensively. In [25], the authors proposed a weak string model, which is a truncated ℓ_2-norm penalized least-squares model for smoothing piecewise signals. Based on the model, a DP-based fast optimization algorithm was also proposed.

The pruned exact linear time (PELT) method [26] is a pruning strategy by which a portion of unnecessary computations can be avoided, and hence, its computational complexity is almost linear in most cases. A similar pruning strategy can be found in [27]. However, PELT approach is dedicated to piecewise constant signals, and optimal segmentation algorithm for high-order polynomial is needed, which is therefore the focus of the current paper.

The paper is organized as follows: in Sec. II, a joint segmentation and fitting model is proposed (Subsec. II-A), followed by a computation of the fitting cost (Subsec. II-B). In Sec.III, an optimization algorithm based on preexisting dynamic programming is presented (Subsec. III-A), followed by its initialization (Subsec. III-C) and implementation issues (Subsec. III-D). In Sec. IV, the proposed method is compared with alternative methods (Subsec. IV-A), analyzed in terms of its computational complexity (Subsec. III-E) and tested on real atomic force microscopy data (Subsec. IV-C). In the appendix, we present a detailed derivation of the iterative calculation of the fitting cost (A) and the usage of Hausdorff distance (B).

II. MODELLING

A. PIECEWISE POLYNOMIALS AND BREAKPOINTS

A piecewise polynomial $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^T$ is a signal with K consecutive pieces (or segments) that are separated by $K-1$ breakpoints (or discontinuities) $v_1, v_2, \ldots, v_{K-1}$ ($v_0 = 0$, $v_K = N$ are also introduced for convenience). The k-th segment ($k = 1, 2, \ldots, K$) is denoted as $\mathcal{I}_k = \{v_{k-1}+1, v_{k-1}+2, \ldots, v_k\}$, which obeys P-th order polynomial:

$$y_i = \sum_{p=0}^{P} c_{p,k}(i - v_{k-1})^p, \quad \forall\, i \in \mathcal{I}_k, \tag{1}$$

where $c_{p,k}$ is the p-th polynomial coefficient of the k-th piece, and the k-th breakpoint is the boundary where the polynomial coefficients of k-th and $k+1$-th pieces change, *i.e.,* $c_{p,k}$ and $c_{p,k+1}$.

Following the framework of analyzing piecewise distributed signals [28], the segmentation and fitting of such kinds of signals can be formulated as

$$\min_{\boldsymbol{C},\mathcal{I}} \left\{ \sum_{k=1}^{K} \varepsilon(\boldsymbol{y}_{\mathcal{I}_k} | \boldsymbol{c}_k) + \lambda K \right\}, \tag{2}$$

where $\boldsymbol{y}_{\mathcal{I}_k}$ is the k-th piece of $\boldsymbol{y}$, *i.e.,* a sub-vector of $\boldsymbol{y}$ with indices $\mathcal{I}_k$, $\boldsymbol{c}_k = [c_{0,k}, c_{1,k}, \ldots, c_{P,k}]^T \in \mathbb{R}^{P+1}$, which is the collection of $P+1$ polynomial coefficients of the k-th piece, $\boldsymbol{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K] \in \mathbb{R}^{(P+1)\times K}$ is the collection of K coefficients vectors, $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_K\}$ is the segmentation scheme, $\varepsilon(\boldsymbol{y}_{\mathcal{I}_k}|\boldsymbol{c}_k)$ is the fitting error (given $\boldsymbol{c}_k$) or the negative log-likelihood chosen as the least-squares fitting error in this paper, and λ is the penalty for each introduction of a piece. Therefore, a larger/smaller λ yields fewer/more segments and hence controls the trade-off between fitting fidelity and smoothness. For a list of strategies on turning λ, interested readers can refer to [28].

In this paper, we refer to term *segmentation* as the estimation of $\mathcal{I}$ and to term *fitting* as the estimation of $\boldsymbol{C}$. We also note that K is defined as the number of pieces (number of breakpoints plus one), so it is a variable that depends on the set $\mathcal{I}$ and is in fact the cardinality of this set. As a result, Eq. (2) can be viewed as a least-squares regression penalized by an ℓ_0-norm, which takes the number of breakpoints into account.

Since the estimation for segmentation ($\mathcal{I}$) and fitting ($\boldsymbol{C}$) are coupled in (2), further simplification is needed. In fact, if the best segmentation $\mathcal{I}$ is known, the estimation of $\boldsymbol{C}$ can be decoupled into K subproblems, and each can be solved separately (which will be shown later in Subsec. II-B). As a result, problem (2) can be reformulated as the following segmentation problem:

$$\min_{\mathcal{I}} \left\{ \sum_{k=1}^{K} \varepsilon(v_{k-1}+1, v_k) + \lambda K \right\}, \tag{3}$$

where $\varepsilon(v_{k-1}+1, v_k)$ is the least-squares fitting error of the k-th segment, whose computation will be presented in the next subsection.

B. COMPUTATION OF THE FITTING ERROR $\varepsilon(u, v)$

1) DIRECT COMPUTATION

Without loss of generality, suppose that a segment starts at u and ends at v, $(v > u+P)$ and is denoted by $z = [y_u, \ldots, y_v]^T \in \mathbb{R}^l$, where $l = v-u+1$ is the length of segment z. For the sake of simplicity, the dependency of k is omitted in the following derivation. The least-squares fitting residual of z with a P-th order polynomial yields:

$$\varepsilon(u, v) \triangleq \varepsilon(z|\boldsymbol{c}) = \|z - \boldsymbol{V}\boldsymbol{c}\|^2 \tag{4}$$

where $\boldsymbol{c} = [c_0, \ldots, c_P]^T \in \mathbb{R}^{P+1}$ is the fitting coefficient vector, and $\boldsymbol{V} \in \mathbb{R}^{l\times(P+1)}$ is a Vandermonde matrix [29] ($[\boldsymbol{V}]_{ij} = i^{j-1}$) which reads

$$\boldsymbol{V} = \begin{bmatrix} 1 & 1 & 1^2 & \cdots & 1^P \\ 1 & 2 & 2^2 & \cdots & 2^P \\ 1 & 3 & 3^2 & \cdots & 3^P \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & l & l^2 & \cdots & l^P \end{bmatrix}. \tag{5}$$

Denoting the inverse Gram matrix as $\boldsymbol{G} = (\boldsymbol{V}^T\boldsymbol{V})^{-1} \in \mathbb{R}^{(P+1)\times(P+1)}$ and $\boldsymbol{\beta} = \boldsymbol{V}^T z \in \mathbb{R}^{P+1}$, it is easy to obtain

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{V}^+ z = \boldsymbol{G}\boldsymbol{\beta} \\ z^* &= \boldsymbol{V}\boldsymbol{c} = \boldsymbol{V}\boldsymbol{G}\boldsymbol{\beta} \\ \varepsilon(u, v) &= \varepsilon(z|\boldsymbol{c}) = z^T z - z^{*T} z^* \\ &= z^T z - \boldsymbol{\beta}^T \boldsymbol{G}\boldsymbol{\beta}, \end{aligned} \tag{6}$$

where $\boldsymbol{V}^+ = (\boldsymbol{V}^T\boldsymbol{V})^{-1}\boldsymbol{V}^T$ is the Moore–Penrose pseudo inverse of $\boldsymbol{V}$.

2) ITERATIVE COMPUTATION

Since the aforementioned direct method computes the ε's separately, and the computation of a single $\varepsilon(u, v)$ needs $(\mathcal{O}(\frac{5}{2}l^2))$ flops [30] due to utilizing the Vandermonde structure, the computational complexity of obtaining all ε's is approximately $\mathcal{O}(\frac{1}{4}N^4)$, $(\sum_{u=1}^{N}\sum_{v=u+P+1}^{N}\frac{5}{2}(v-u+1)^2)$, which is burdensome. In this paper, we introduce a speedup method, and the idea is to calculate $\varepsilon(u, v+1)$ based on $\varepsilon(u, v)$ by iteratively updating a few matrices and vectors. As a result, the total computational complexity is reduced to $\mathcal{O}(\frac{3}{2}(P+2)^2N^2)$ for the worst case. The detailed mathematical derivation is shown in Appendix A, and the detailed computational complexity is shown in Subsec. III-E.

III. OPTIMIZATION ALGORITHM

A. PREEXISTING ALGORITHMS

With all fitting errors ε in hand, the optimization problem (3) involves finding the best segmentation among all possibilities. A well-known search method is so-called dynamic programming [31], which solves the following problem iteratively [22], [23]:

$$\phi(v+1) = \min_{1\leqslant u\leqslant v} [\phi(u) + \varepsilon(u, v)] + \lambda, \tag{7}$$

where $\phi(v)$ is the minimal criterion value of only the first $v-1$ points of $\boldsymbol{y}$ under the assumption that $v-1$ is a breakpoint. The pseudo code of this method is shown in Tab. 1 as Algorithm I, where u^* is the optimizer of problem (7), q is a set of vectors, with $q\{v\}$ a vector which stores the obtained breakpoints of only the first $v-1$ points of $\boldsymbol{y}$ under the assumption that $v-1$ is a breakpoint. The outputs $\phi(N+1)$ and $q\{N+1\}-1$ are the optimal objective value and set of breakpoints of problem (3), respectively.

Algorithm I provides the optimal solution to this problem, but since it explores all $\varepsilon(u, v)$ with $u+P < v$, the number of ε to be computed is quadratic with respect to the signal length N. Early works discovered that under a specific condition (*e.g.,* Theorem 3.1 of [26], Algorithm 3 of [27]), some indices u are *inutile* and hence are *pruned* to avoid computing large amounts of $\varepsilon(u, v)$ in successive iterations. The benefit of this discovery is that the optimality of the solution is maintained meanwhile the efficiency of computation is improved. To incorporate this pruning strategy, a set $\mathcal{S}$ is introduced to store only these *utile* indices:

$$\mathcal{S} = \{u | 1 \leqslant u \leqslant v, \phi(u) + \varepsilon(u, v) < \phi(v+1)\}. \tag{8}$$

As a result, the exploration space of ε in each iteration (problem (7)) is reduced from $1 \leqslant u \leqslant v$ to $u \in \mathcal{S}$, which may involve up to linear complexity with respect to the signal length N in many cases [26], [27]. The pseudocode of this improved method is shown in Tab. 2 as Algorithm II. In fact, if the pruning step is disabled in Algorithm II, or the insert step is modified to $\mathcal{S} = [1 : v+1]$), Algorithm II degenerates

TABLE 1. Algorithm I: basic dynamic programming.

```
Input: $\varepsilon, P, \lambda$.
Initialize $\phi(1:P+1) = [0, \lambda, \ldots, \lambda], q\{1\} = [\,], q\{2\} = [1], \ldots,$
          $q\{P+1\} = [P]$
For $v = P+1 : N$
          solve $[\phi(v+1), u^*] = \min_{1\leqslant u\leqslant v} [\phi(u) + \varepsilon(u, v)] + \lambda$
          store $q\{v+1\} = [q\{u^*\}, u^*]$
End
Output: $\phi(N+1), q\{N+1\} - 1$.
```

TABLE 2. Algorithm II: Algorithm I with pruning.

```
Input: $\varepsilon, P, \lambda$.
Initialize $\phi(1:P+1) = [0, \lambda, \ldots, \lambda], q\{1\} = [\,], q\{2\} = [1], \ldots,$
          $q\{P+1\} = [P], \mathcal{S} = [1 : P+1]$
For $v = P+1 : N$
          solve $[\phi(v+1), u^*] = \min_{u\in\mathcal{S}} [\phi(u) + \varepsilon(u, v)] + \lambda$
          store $q\{v+1\} = [q\{u^*\}, u^*]$
          prune $\mathcal{S} = [u | u \in \mathcal{S}, \phi(u) + \varepsilon(u, v) < \phi(v+1)]$
          insert $\mathcal{S} = [\mathcal{S}, v+1]$
End
Output: $\phi(N+1), q\{N+1\} - 1$.
```

TABLE 3. **Algorithm III: modified initialization and definition of ε.**

```
Input: $\varepsilon, P, \lambda$.
Initialize $\phi(1) = [0], q\{1\} = [\,], \mathcal{S} = [1], s = 1$
For $v = P + 1 : N$
    modify $\bar{\mathcal{S}} = md(\mathcal{S})$ according to (9)
    solve $[\phi(v - P + 1), u^*] = \min_{1 \leqslant u \leqslant s} [\phi(\bar{\mathcal{S}}(u)) + \varepsilon(\mathcal{S}(u), v)] + \lambda$
    store $q\{v - P + 1\} = [q\{\bar{\mathcal{S}}(u^*)\}, \bar{\mathcal{S}}(u^*)]$
    prune $\mathcal{S} = [\mathcal{S}(u) | 1 \leqslant u \leqslant s, \phi(\bar{\mathcal{S}}(u)) + \varepsilon(\mathcal{S}(u), v) < \phi(v - P + 1)]$
    insert $\mathcal{S} = [\mathcal{S}, v - P + 1], s = |\mathcal{S}|$
End
Output: $\phi(N - P + 1), q\{N - P + 1\} + P - 1$.
```

TABLE 4. **Algorithm IV: HOPS.**

```
Input: $y, P, \lambda$.
Initialize $\phi(1) = [0], q\{1\} = [\,], \mathcal{S} = [1], B(1) = [V_0^T y(1 : 1 + P)], s = 1, \Omega(1) = [G_0], e(1) = [0]$
For $v = P + 1 : N$
    modify $\bar{\mathcal{S}} = md(\mathcal{S})$ according to (9)
    solve $[\phi(v - P + 1), u^*] = \min_{1 \leqslant u \leqslant s} [\phi(\bar{\mathcal{S}}(u)) + e(u)] + \lambda$
    store $q\{v - P + 1\} = [q\{\bar{\mathcal{S}}(u^*)\}, \bar{\mathcal{S}}(u^*)]$
    prune $\mathcal{S} = [\mathcal{S}(u)], \Omega = [\Omega(u)], e = [e(u)], B = [B(u)], \forall\{u | 1 \leqslant u \leqslant s, \phi(\bar{\mathcal{S}}(u)) + e(u) < \phi(v - P + 1)\}$
    compute $A$, $\Gamma$, and $\rho$ according to (10), (13), and (12), respectively.
    update $B$, $\Omega$, and $e$ according to (11), (14), and (15), respectively.
    insert $\mathcal{S} = [\mathcal{S}, v - P + 1], s = |\mathcal{S}|, B = [B, V_0^T y(v - P + 1 : v + 1)], \Omega = [\Omega, G_0], e = [e, 0]$
End
Output: $\phi(N - P + 1), q\{N - P + 1\} + P - 1$.
```

to Algorithm I; if $P = 0$, Algorithm II degenerates to the PELT method.

To clarify the amount of ε being explored, let us define a matrix $\boldsymbol{E} = [\varepsilon(u, v)] \in \mathbb{R}^{N \times N}$, where $\varepsilon(u, v)$ is the least-square fitting error from data point y_u to y_v, and has been calculated explicitly at the termination of execution.

Several typical $\boldsymbol{E}$'s are demonstrated in Fig. 3, in which black is used to illustrate the elements of $\boldsymbol{E}$ being calculated. Since Algorithm I explores all $\varepsilon(u, v)$ with $u + P < v$, the black region of $\boldsymbol{E}$ is an upper triangle. For the PELT method, if $\varepsilon(u_0, v_0)$ violates the condition in (8), u_0 is removed from set $\mathcal{S}$, implying that it is not necessary to compute $\varepsilon(u_0, v)$ for $v > u_0$, forming a horizontal white line. As a result, the black region obtained by the PELT method is a pileup of irregular horizontal lines, which are effective computations.

B. PROPOSED ALGORITHM

Here we introduce the proposed high-order polynomial segmenter (HOPS), which uses dynamic programming to solve problem (3). The HOPS combines both the iterative computation of ε and the pruning strategy of $\mathcal{S}$, and hence it can segment piecewise polynomials of arbitrary order efficiently. The pseudo code of the HOPS is listed in Tab. 4.

Subsec. II-B and Appendix A provide the detailed derivation of the iterative computation of $\varepsilon(u, v + 1)$ based on $\varepsilon(u, v)$. According to (25), $\varepsilon(u, v + 1)$ depends on $\varepsilon(u, v)$ from the previous iteration, the new data point y_{v+1}, and the three intermediate variables: ρ, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$. The last three variables further depend on $\boldsymbol{\alpha}$ and $\boldsymbol{G}$, which are described as follows. (a) according to (16), $\boldsymbol{\alpha}$ depends only on the length of the analyzed segment; (b) according to (17), $\boldsymbol{\gamma}$ depends on $\boldsymbol{G}$ from the previous iteration and the value of $\boldsymbol{\alpha}$ computed in (16); (c) according to (18), ρ depends on the value of $\boldsymbol{\alpha}$ computed in (16) and that of $\boldsymbol{\gamma}$ from (17); (d) according to (21), $\boldsymbol{\beta}$ depends on $\boldsymbol{\beta}$ from the previous iteration, the value of $\boldsymbol{\alpha}$ computed in (16), and the new data point y_{v+1}; (e) according to (22), $\boldsymbol{G}$ depends on the previous iteration, the value of $\boldsymbol{\gamma}$ computed in (17), and the value of ρ from (18).

C. INITIALIZATION

The initialization process is nontrivial when $P \neq 0$. According to the common definition of $\phi(v)$ (the minimal criterion value for the first $v - 1$ points), $\phi(1 : P + 1)$ is initialized with $\phi(1 : P + 1) = [0, \lambda, \ldots, \lambda]$ as shown in Tabs. 1 and 2. q and $\mathcal{S}$ are initialized in the same manner. This initialization method makes sense but is redundant.

In fact, if we redefine $\phi(v)$ as the minimal criterion value of the first $v - 1 + P$ points, an alternative and more compact initialization method is $\phi(1) = 0, q\{1\} = [\ ], \mathcal{S} = [1]$. Tab. 3 shows the modified pseudocode of Tab. 2 with an alternative initialization and definition of ϕ. Note that since the alignment of ϕ and ε is destroyed, the index of ϕ ($\mathcal{S}$ and q)) should be modified by following a pointwise function 'md,' which is defined as

$$md(t) = \begin{cases} 1, & t = 1; \\ 2, & 2 \leqslant t \leqslant P + 1; \\ t - P, & t > P + 1. \end{cases} \quad (9)$$

Note that when $P = 0$, both initialization methods are identical.

D. IMPLEMENTATION

In each iteration of the proposed algorithm, a batch of ε's (a column in $\boldsymbol{E}$) are calculated. For each ε, five intermediate variables $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$, ρ, $\boldsymbol{\beta}$, and $\boldsymbol{G}$ should be computed or loaded from the previous iteration. Therefore, an efficient data structure is needed to store these variables for further operation.

More specifically, to calculate $\varepsilon(u, v+1)$ from $\varepsilon(u, v)$, $\forall u \in \mathcal{S}$, Appendix A shows that six sets (five for the intermediate variables, plus one for ε) are needed, and each set has s elements, where $|\cdot|$ is the cardinality of a set. Therefore, we introduce four matrices $\boldsymbol{A} \in \mathbb{R}^{(P+1)\times s}$, $\boldsymbol{\Gamma} \in \mathbb{R}^{(P+1)\times s}$, $\boldsymbol{B} \in \mathbb{R}^{(P+1)\times s}$, and $\boldsymbol{\Omega} \in \mathbb{R}^{(P+1)^2\times s}$ and two vectors $\boldsymbol{e} \in \mathbb{R}^s$ and $\boldsymbol{\rho} \in \mathbb{R}^s$ to store all the elements of the sets of $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$, $\boldsymbol{G}$, ε, and ρ in Appendix A, respectively. Note that each column of $\boldsymbol{\Omega}$ is a column-wise vectorized version of matrix $\boldsymbol{G}$.

Here $\boldsymbol{A}$ is a Vandermonde matrix, whose columns can be calculated simply from (16), and the segment length l of each column is calculated from the elements in $\mathcal{S}$ and v. More specifically,

$$\boldsymbol{A} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_s], \tag{10}$$

where $\boldsymbol{\alpha}_j = [1, l_j, l_j^2, \ldots, l_j^P]^T, 1 \leqslant j \leqslant s$, and $l_j = v - \mathcal{S}(j) + 3$.

According to (21), updating of $\boldsymbol{B}$ reads

$$\boldsymbol{B}_+ = \boldsymbol{B} + y_{v+1}\boldsymbol{A}. \tag{11}$$

According to (18), calculation of $\boldsymbol{\rho}$ reads

$$\boldsymbol{\rho} = \mathbf{1}_s \oslash (\mathbf{1}_s + (\boldsymbol{\Gamma}^T \odot \boldsymbol{A}^T)\mathbf{1}_{P+1}), \tag{12}$$

where $\oslash$ and $\odot$ represent pointwise division and multiplication (the Khatri-Rao product); $\mathbf{1}_s$ and $\mathbf{1}_{P+1}$ are all-one vectors of length s and $P+1$, respectively.

The computation of $\boldsymbol{\Gamma}$ is complicated, and it reads as

$$\boldsymbol{\Gamma} = sum(\boldsymbol{\Omega} \odot (\boldsymbol{A} \otimes \mathbf{1}_{P+1})), \tag{13}$$

where $\otimes$ is the Kronecker product. This formulation should be understood as the following steps: first, according to (17) we calculate $\boldsymbol{\Omega} \odot (\boldsymbol{A} \otimes \mathbf{1}_{P+1})$, yielding a matrix of size $(P+1)^2 \times s$; then, we reshape this matrix to a $(P+1) \times (P+1) \times s$ three way tensor; and finally, we sum the tensor across the second dimension to obtain the matrix $\boldsymbol{\Gamma}$ of size $(P+1) \times s$.

According to (22), the updating of $\boldsymbol{\Omega}$ reads

$$\boldsymbol{\Omega}_+ = \boldsymbol{\Omega} - [(\boldsymbol{\Gamma} \otimes \mathbf{1}_{P+1})diag(\boldsymbol{\rho})] \odot (\mathbf{1}_{P+1} \otimes \boldsymbol{\Gamma}), \tag{14}$$

where $diag(\boldsymbol{\rho})$ is the diagonal matrix of vector $\boldsymbol{\rho}$.

At last, according to (25), updating of $\boldsymbol{e}$ reads

$$\boldsymbol{e}_+ = \boldsymbol{e} + \boldsymbol{\rho} \odot (y_{v+1}\mathbf{1}_s - (\boldsymbol{\Gamma}^T \odot \boldsymbol{B}^T)\mathbf{1}_{P+1})^2, \tag{15}$$

where the square should be understood as pointwise operation.

The pseudocode of the HOPS is shown in Tab. 4. V_0 is the initial matrix of $\boldsymbol{V}$, *i.e.,* $l = P+1$ in (5), so it is a definition of the Vandermonde matrix $\boldsymbol{V}$; G_0 is the initial vector of matrix $\boldsymbol{G}$, *i.e.,* the vectorized version of $\boldsymbol{G}$ with $l = P+1$. The outputs $\phi(N-P+1)$ and $q\{N-P+1\}+P-1$ are the optimal objective value and set of breakpoints of problem (3), respectively.

TABLE 5. The computational complexity of variables in an iteration of HOPS algorithm.

variable	equation	complexity
$\boldsymbol{A}$	(10)	$(P+1)s$
$\boldsymbol{\Gamma}$	(13)	$(P+1)^2 s$
$\boldsymbol{\rho}$	(12)	$(P+1)s$
$\boldsymbol{B}$	(11)	$(P+1)s$
$\boldsymbol{\Omega}$	(14)	$2(P+1)^2 s$
$\boldsymbol{e}$	(15)	$(P+1)s$
total	\	$(3P+7)(P+1)s$

In summary, since the HOPS algorithm relies on a set $\mathcal{S}$ to prune unnecessary calculations of ε, and uses four matrices and two vectors to execute the necessary calculation of ε iteratively, one can achieve improved computational efficiency.

E. COMPUTATIONAL COMPLEXITY

Since the above pruning strategy is incorporated into the HOPS, the computational complexity varies according to the number of breakpoints and the value of λ. The best case occurs when breakpoints are everywhere, or λ is small; the worst case occurs when no breakpoint is detected, or λ is large. In the best case, pruning occurs during each iteration and the set $\mathcal{S}$ is a singleton; hence the cardinality s is always 1. In the worst case, no index is pruned from the set $\mathcal{S}$, and hence the cardinality s increases from 1 to $N-P$.

Tab. 5 lists the computational complexities for computing the variables in an iteration ($|\mathcal{S}| = s$) of the HOPS algorithm. Hence, in the worst case, the total computational complexity of the HOPS is upper $\sum_{s=1}^{N-P}[3(P+1)+4](P+1)s \approx \frac{3}{2}(P+2)^2N^2$, which is quadratic with respect to the signal length N. In the best case the computational complexity is $3(P+2)^2N$, which is linear with respect to N.

IV. RESULTS

A. SEGMENTATION PERFORMANCE

1) REPRESENTATIVE METHODS

Since many methods have been developed to segment piecewise polynomial signals, we selected two representatives, one for $P=0$ and another for $P=1$:

- The ℓ_1-penalized total variation [2], [32], is a hybridized model that utilizes total variation [12] to segment and fit piecewise constant signals. The model was first transformed into a standard LASSO problem by the formulae in [33], and then solved by the LASSO solver *solveLasso* of *SparseLab* at http://sparselab.stanford.edu/. In the following simulations, the results of this method are labeled with *L1TV*.
- The ℓ_1 trend filter by Kim *et al.* [7], uses the ℓ_1-norm to penalize the second-order differences for recovering piecewise linear signals. The package *l1_tf* was downloaded from https://web.stanford.edu/~boyd/l1_tf/, and it is an MATLAB implementation of the interior-point method. In the following simulations, the results of this method are labeled with *L1TF*.

2) SIMULATION I: PIECEWISE CONSTANT SIGNALS ($P = 0$)

This simulation aimed to test the performance of the HOPS with regard to the segmentation of piecewise constant signals ($P = 0$) and to compare it with that of L1TV. Since the HOPS extends the PELT method to high-order polynomials, both always obtained the same results in this simulation, yielding overlapping curves; hence, the results of the PELT method are not shown.

First, a piecewise constant signal of length $N = 1000$ was simulated, with break points following a Bernoulli-Gaussian distribution [34], *i.e.,* the Bernoulli probability of a breakpoint was 0.01, and its amplitude followed a normal distribution. Specifically, i.i.d. Gaussian noise was added, with mean 0 and standard deviation 0.1. A typical signal is shown in Fig. 1(b) as a black dotted curve.

Then, the signal was segmented with both the HOPS and L1TV methods, with 9 values of penalty parameters from 1e-2 to 1e2 with common ratio of 1e 0.5. Typical fitting and detection results are demonstrated in Fig. 1(b) and (c), with 1e-0.5 as the value of the penalty parameter. This example shows that both the HOPS and L1TV reconstruction preserved the piecewise characteristics of the original data, but L1TV tended to yield several minor breakpoints with small amplitudes.

To quantitatively evaluate the segmentation performance, the Hausdorff distances between the ground-truth breakpoints and detected breakpoints were evaluated. For the detailed calculation of the Hausdorff distance, readers are referred to Appendix B. Fig. 1 (a) shows the average Hausdorff distance for 100 Monte-Carlo replications with respect to the penalty parameters. It is shown that the HOPS achieved better performances than L1TV in terms of breakpoint detection when the penalty parameter was not too small. Panels (b) and (c) are the results corresponding to the lowest points (starts) in Panel (a).

3) SIMULATION II: PIECEWISE LINEAR SIGNALS ($P = 1$)

In this subsection we tested the performance of HOPS on piecewise linear signals, and compared it with that of the ℓ_1 trend filter developed by Kim *et al.* [7].

One hundred piecewise linear signals whose length is 1000 were simulated, and breakpoints were distributed uniformly along the loci with a Bernoulli probability of 0.005. For each segment of a signal, both the intercept and the slope followed a Gaussian distribution; for each signal, i.i.d. Gaussian noise was added. All three Gaussian distribution (intercept, slope, and noise) had means of zero, and they had standard deviations of 1, 0.001, and 1, respectively.

The 100 simulated piecewise linear signals were processed with the HOPS and ℓ_1 trend filter following the same process as in Simulation I, and the results are shown in Fig. 2. Panel (a) shows that the HOPS achieved slightly better performances than those of the ℓ_1 trend filter in terms of the Hausdorff distance metric, and similar performances in terms

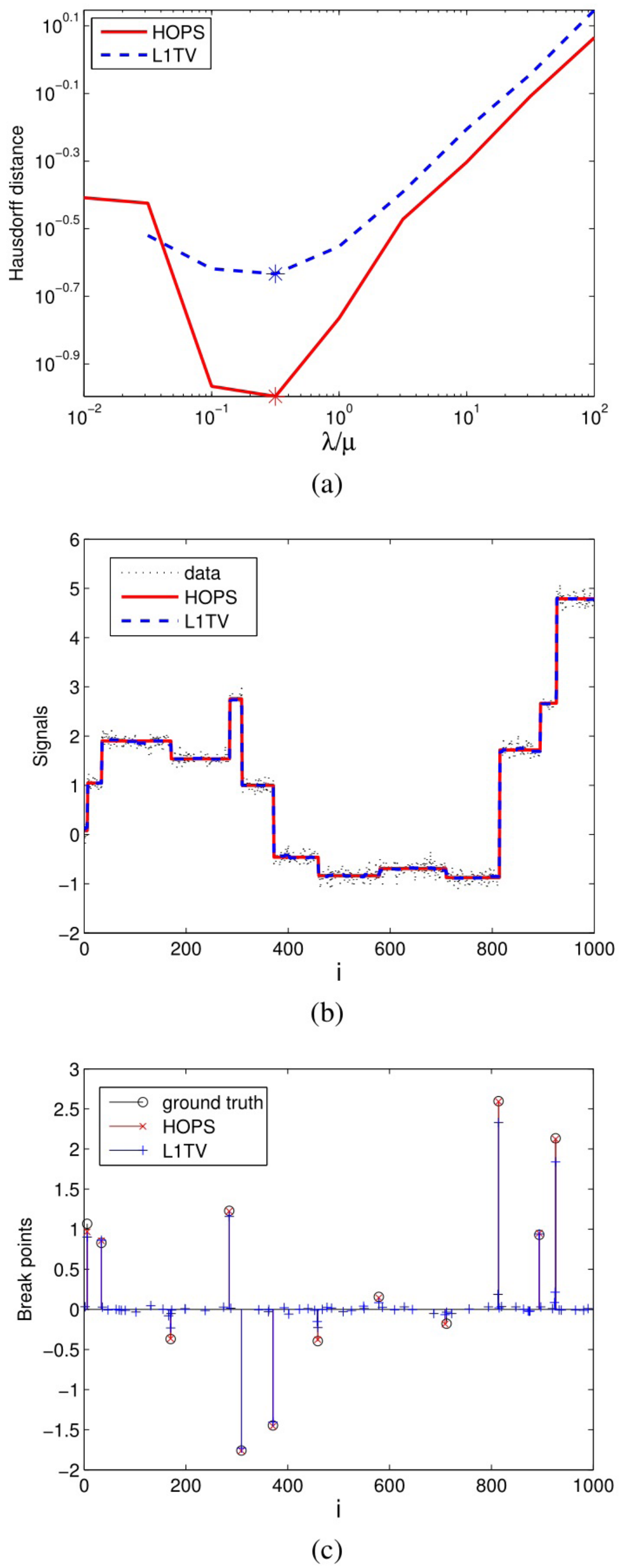

FIGURE 1. Results of Simulation I: piecewise constant signals ($P = 0$). (a) is the average Hausdorff distance with respect the penalty parameters (λ of HOPS, and μ of the L1TV). (b) demonstrates a typical signal and its recoveries at the stars in (a). (c) demonstrates the corresponding breakpoints detected.

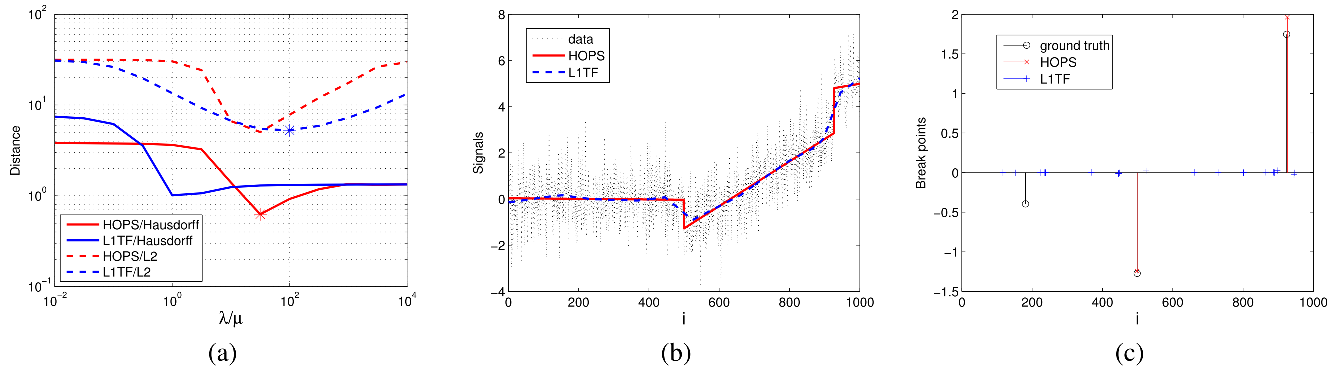

FIGURE 2. Results of Simulation II: piecewise linear signals ($P = 1$). (a) is the average Hausdorff and Euclidean distance with respect the penalty parameters (λ of HOPS, and μ of L1TF). (b) demonstrates a typical signal and its recoveries at the stars in (a). (c) demonstrates the corresponding breakpoints detected.

of Euclidean distance. The blue star is the lowest point on Euclidean distance curve instead of on the Hausdorff distance curve, providing larger penalty parameters and fewer breakpoints. The corresponding signals and breakpoints are displayed in panels (b) and (c). The conclusion is the same as that in Simulation I: the ℓ_0 norm can achieve higher sparsity level for the breakpoints than that obtained using the ℓ_1 norm, which tends to detect more breakpoints with smaller amplitude.

B. COMPUTATIONAL PERFORMANCE

In this subsection, comprehensive studies were conducted to test the computational performance of the HOPS. The Matlab codes were run on a desktop with an Intel i7-3770 processor and 32 GB of memory. The stopwatch commands ‘tic’ and ‘toc’ were used to record the execution time.

1) SIMULATION III: COMPUTATIONAL BEHAVIOR OF $\boldsymbol{E}$

First, a signal $\boldsymbol{y}$ of length 100 with only i.i.d. Gaussian noise was generated, and each point obeyed a normal distribution (zero mean and unit variance). Then, this signal was processed with $P = 0$ and $\lambda = 100$, which is a relatively large penalty such that no breakpoint should have been detected. Fig. 3 panel (a) demonstrates the elements of matrix $\boldsymbol{E}$ being explicitly calculated (in black). Note that since no pruning event occurred, $\boldsymbol{E}$ wan an upper triangle.

Second, a breakpoint of amplitude 5 was inserted at location 50, yielding $\boldsymbol{y}$, which is a step signal whose first (y_1 to y_{50}) and second (y_{51} to y_{100}) halves had means of 0 and 5, respectively. $\boldsymbol{y}$ was also processed with $\lambda = 100$, and $\boldsymbol{E}$ is shown in penal (b). Note, since a pruning event happened at location 50, a white square appeared, and hence the computational burden was reduced by almost half.

A breakpoint of amplitude -10 was further inserted at location 80, and the resultant $\boldsymbol{E}$ is shown in panel (c). It is shown that the computational burden was further reduced by a quarter. Panels (d) and (e) used the same data as panels (a) and (b) respectively, but with $\lambda = 1$, which is a relatively small penalty value such that breakpoints were detected almost everywhere. Note that (d) and (e) are close, but not the same. In addition, (f) used the same data from (c) but with $\lambda = 10$ to demonstrate the evolution of $\boldsymbol{E}$ with respect to λ. We conclude that as λ increases (or the number of breakpoints decreases), the computational burden increases.

2) SIMULATION IV: COMPUTATION TIME WITH RESPECT TO THE BREAKPOINT PERCENTAGE

The study of $\boldsymbol{E}$’s behavior showed that the number of breakpoints greatly influences the computation time, hence in this test breakpoints were inserted into random Gaussian signal. The simulation was generated following the same approach as in the previous simulation (zero mean, unit variance). Breakpoints were inserted as follows: (1) the number of breakpoints was calculated from the length of the signal N and the percentage of breakpoints (1%,0.5%, and 0.1% for the three sparsity scenarios); (2) the locations and amplitudes of the breakpoints were generated, where each location obeyed a uniform distribution from 1 to N, and the amplitude obeyed a standard normal distribution; (3) each breakpoint was inserted into the random Gaussian signal according to the obtained locations and amplitudes.

Ten Monte-Carlo replicates were processed in the same way as in the previous simulation ($\lambda = 100$, $P = 0$), and the segmentation results are shown in Fig. 4 (a). It is shown that as the number of breakpoints increases, the computational time decreases greatly, especially for long signals. From this figure, we can estimate that the computational time of a signal with millions of points and thousands of breakpoints is approximately three hours. This is helpful for the segmentation of long signals such as DNA sequencing data.

3) SIMULATION V: COMPUTATIONAL TIME IN THE WORST CASE

We also tested the computational time required in the worst case for the proposed HOPS algorithm, with respect to the polynomial order P and signal length N.

In this test, 10 simulated signals were generated for each configuration of N and P; this was done in the same way as in Simulation II. λ was set to 100 to mimic the upper bound computation.

The results are shown in Fig. 4 (b), which shows that (1) the logarithm of the computational time increases linearly

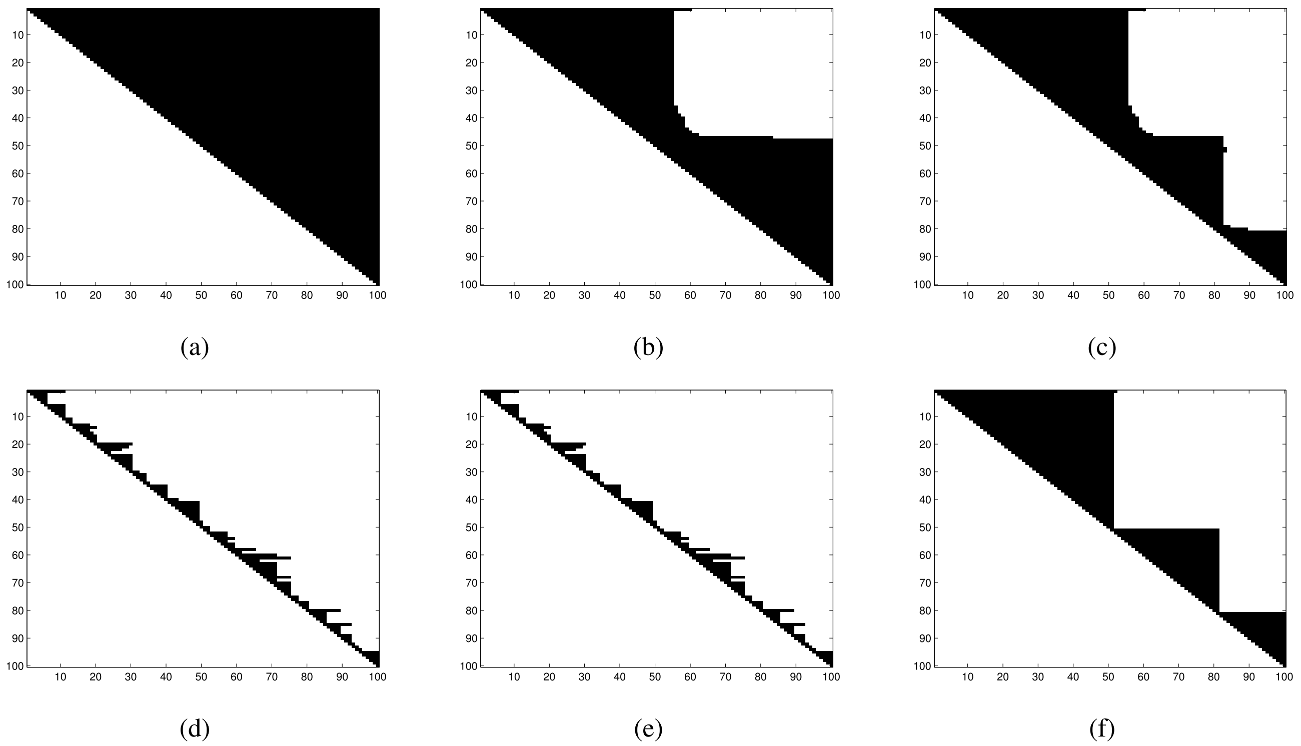

FIGURE 3. Results of Simulation III: computational behavior of matrix E ($N = 100$). The black color indicates that the element $\varepsilon(u, v)$ of E is computed. (a) and (d) use the same data y with no breakpoint. (b) and (e) use the same data with a breakpoint at 50. (c) and (f) use the same data with two breakpoints at 50 and 80. λ of (a) (b) and (c) is 100, (d) and (e) is 1, (f) is 10.

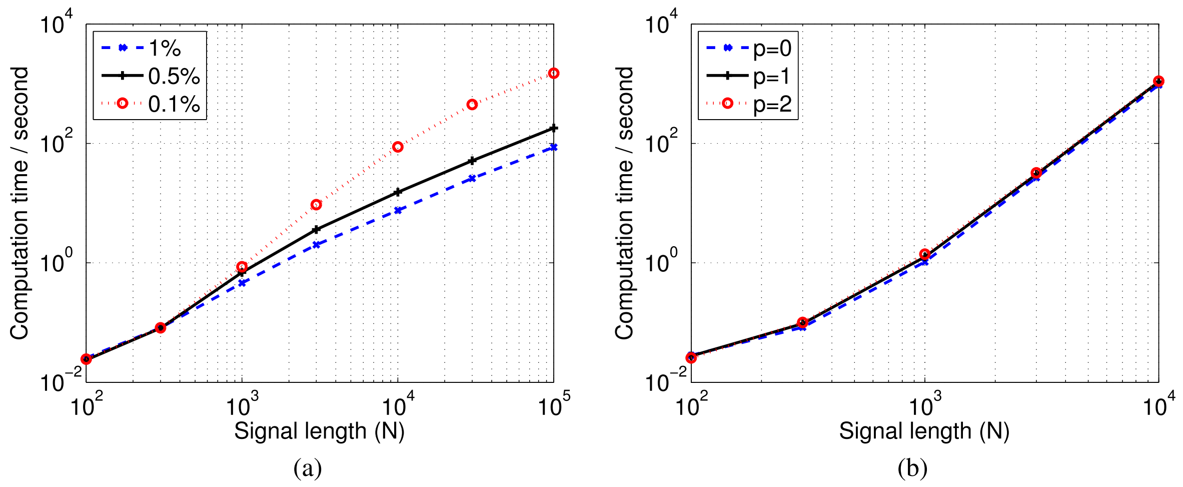

FIGURE 4. Results of Simulation IV and V. (a) computation time with respect to signal length N and percentage of breakpoints. (b) the computation time with respect to signal length N and polynomial order P in the worst case.

as the signal length increases with slope of approximately 2, indicating quadratic computational complexity in the worst case; (2) the polynomial order P has limited influence on the computational time.

C. REAL DATA PROCESSING: AFM FORCE CURVES OF UNFOLDING PROTEINS

Atomic force microscopy (AFM) is a high-resolution scanning probe microscopy [4]. The data acquired by AFM is a set of force curves $f(z)$, where f is the interaction force between the probe and sample, and z is the indentation of the piezoelectric scanner related to the distance between the probe and sample. By analyzing the force curves, important properties of the samples, such as Young's modulus, can be investigated [35]. Due to its high-resolution (usually up to nanometer and nano Newton levels), AFM is widely utilized in nanotechnology, biotechnology, *etc.*

An insightful application of AFM is to investigate protein unfolding [36]. Here, we used AFM data sampled on Xenopus laevis oocytes, to study the cyclic nucleotide-gated channel subunit alpha 1 (CNGA1) [37]. CNGA1 consists of secondary structure units, and when a stretching force

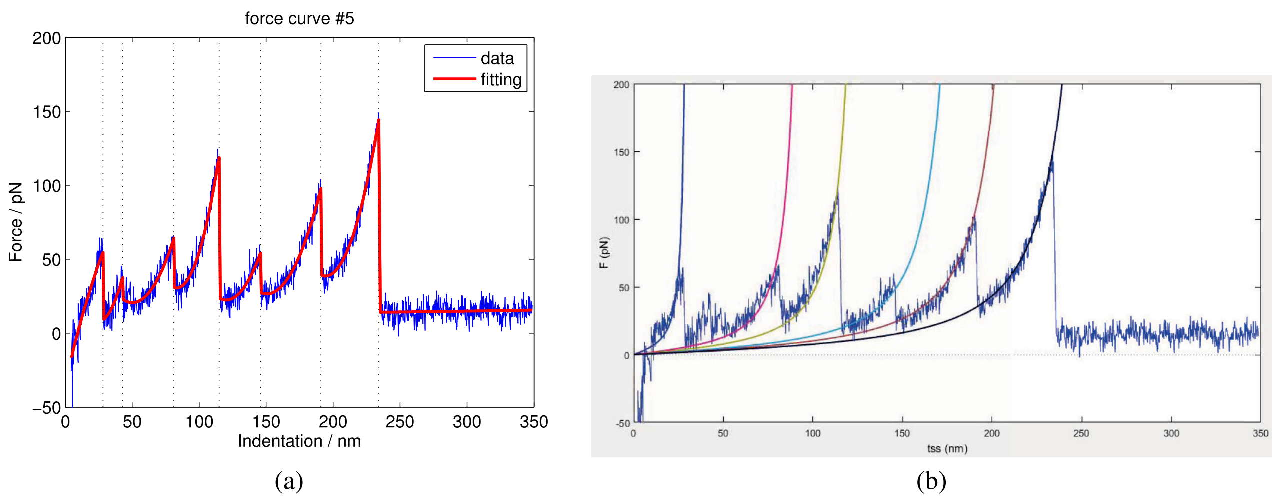

FIGURE 5. Results of real data processing: Segmenting and fitting of a protein unfolding AFM force curves [37]. (a) is the result of HOPS with $P = 2$, (b) is the result of Fodis [36].

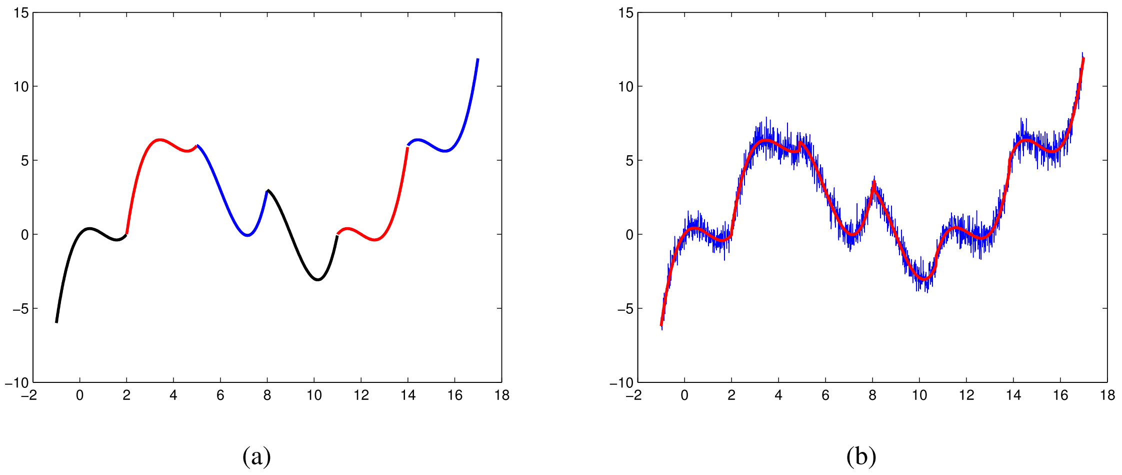

FIGURE 6. Results of high order performance ($P = 3$). (a) the noiseless signal with six pieces of cubic polynomials, (b) signal with noise (blue) and fitting (red), detected breakpoints are 2.06, 5.08, 8.05, 11.05 and 14.05.

is applied, the conformation changes. As a result, AFM retraction force curves contain several breakpoints, and each segment between two neighboring discontinuities can be modeled with a worm-like chain (WLC) or freely jointed chain (FJC) [4]. By fitting these segments, structural parameters, such as the contour length and persistence length can be estimated.

Several methods have been proposed to segment this kind of data [35]–[38], but they are still not automatic and contain several parameters that need to be turned manually. Here we show that the HOPS can provide great results, with only two parameters: P and λ. The AFM force curve data was download at https://github.com/nicolagalvanetto/Fodis/blob/master/Datasets/ TXT%20universal%20import%20files/104_CNGA1_selected.txt, and the 9th and 10th columns were extracted as f and z, respectively, which are shown as the blue curve in Fig. 5(a). After preprocessing [38], the force curve was processed with the HOPS, and the result is shown as the red curve. The detected 7 breakpoints are displayed with vertical dashed lines. In this analysis, polynomials with order $P = 2$ and $\lambda = 2e3$, were used. As a comparison, the result of Fodis [36] is shown in panel (b), for which the breakpoint near 50 nm is cumbersome.

D. HIGH ORDER PERFORMANCE

Previous experiments demonstrated the performance of HOPS with low order polynomials ($P \leqslant 2$), hence in the last experiment, we showcase a high order example ($P = 3$).

First, a noiseless signal with six pieces of cubic polynomials (Fig. 6(a)) was generated:

$$y(x) = \begin{cases} x(x-1)(x-2), & x \in [-1, 2) \\ (x-3)(x-4)(x-5)+6, & x \in [2, 5) \\ (x-4)(x-6)(x-8)+3, & x \in [5, 8) \\ (x-7)(x-9)(x-11), & x \in [8, 11) \\ (x-11)(x-12)(x-13), & x \in [11, 14) \\ (x-14)(x-15)(x-16)+6, & x \in [14, 17) \end{cases}$$

Second, white Gaussian noise was added to mimic signal-to-noise ratio of 6 dB (blue curve in Fig. 6(b)). Then the signal was processed with the proposed HOPS algorithm, with $P = 3, \lambda = 10$. The fitted curve is shown in Fig. 6(b) as the red curve, with detected breakpoints 2.06, 5.08, 8.05, 11.05 and 14.05, which is pretty close to the ground-truth.

V. CONCLUSION AND DISCUSSION

In this paper, we introduce the high-order polynomial segmenter (HOPS), a fast algorithm for segmenting piecewise polynomials. The proposed algorithm was tested on both simulated and real datasets, and the results support the potential of the HOPS for a broad spectrum of applications.

Several highlights of the HOPS include: (i) it performs the joint segmentation and fitting of piecewise polynomials of arbitrary order; (ii) global optimization is guaranteed; (iii) the computational complexity is linear in most cases. In the worst case, the computational complexity is upper-bounded by quadratic.

Since the HOPS extends the pruned exact linear time (PELT) method [26] to the segmentation of polynomials of arbitrary order, it maintains the same computational advantages as those of the PELT approach. As noted in [26], although the worst-case complexity is $\mathcal{O}(N^2)$, in most cases the complexity is 'linear time.' Our analysis (iii) in the last paragraph agrees with this assessment. Furthermore, HOPS uses matrix factorization to accelerate the calculation of ε, and hence, it can segment long signals with high efficiency.

Finally, as general guidance for piecewise signal fitting and segmentation, if the average distance between breakpoints is below $1e4$, the HOPS is recommended. If computational speed is the main concern (within seconds) for long signals ($N > 1e6$) with a handful of breakpoints, ℓ_1-based methods or other suboptimal methods, *e.g.,* CBS, are recommended.

APPENDIX A
ITERATIVE COMPUTATION OF $\varepsilon(u, v+1)$ BASED ON $\varepsilon(u, v)$

For a segment of length $l + 1 = v - u$, denote

$$\boldsymbol{\alpha} = [1, l+1, (l+1)^2, \ldots, (l+1)^P]^T, \quad (16)$$

$$\boldsymbol{\gamma} = \boldsymbol{G}\boldsymbol{\alpha}, \quad (17)$$

$$\rho = (1 + \boldsymbol{\alpha}^T\boldsymbol{\gamma})^{-1}, \quad (18)$$

$$z_+ = \left[\frac{z}{y_{v+1}}\right] \in \mathbb{R}^{l+1}, \quad (19)$$

$$\boldsymbol{V}_+ = \left[\frac{\boldsymbol{V}}{\boldsymbol{\alpha}^T}\right] \in \mathbb{R}^{(l+1)\times(P+1)}, \quad (20)$$

so

$$\begin{aligned}\boldsymbol{\beta}_+ &= \boldsymbol{V}_+^T z_+ = \left[\boldsymbol{V}^T \middle| \boldsymbol{\alpha}\right]\left[\frac{z}{y_{v+1}}\right] \\ &= \boldsymbol{V}^T z + y_{v+1}\boldsymbol{\alpha} \\ &= \boldsymbol{\beta} + y_{v+1}\boldsymbol{\alpha}. \quad (21)\end{aligned}$$

From Woodbury's matrix identity, the augmented Gram inverse matrix reads

$$\begin{aligned}\boldsymbol{G}_+ &= (\boldsymbol{V}_+^T\boldsymbol{V}_+)^{-1} \\ &= (\boldsymbol{V}^T\boldsymbol{V} + \boldsymbol{\alpha}^T\boldsymbol{\alpha})^{-1} \\ &= \boldsymbol{G} - \rho\boldsymbol{\gamma}\boldsymbol{\gamma}^T. \quad (22)\end{aligned}$$

So from (6) the augmented fitting residual reads

$$\begin{aligned}\varepsilon(u, v+1) &= z_+^T z_+ - \boldsymbol{\beta}_+^T\boldsymbol{G}_+\boldsymbol{\beta}_+ \quad (23)\\ &= z_+^T z_+ - (\boldsymbol{\beta} + y_{v+1}\boldsymbol{\alpha})^T \\ &\quad \cdot(\boldsymbol{G} - \rho\boldsymbol{\gamma}\boldsymbol{\gamma}^T)(\boldsymbol{\beta} + y_{v+1}\boldsymbol{\alpha}) \\ &= z_+^T z_+ - [\boldsymbol{\beta}^T\boldsymbol{G}\boldsymbol{\beta} - \rho(\boldsymbol{\beta}^T\boldsymbol{\gamma})^2 \\ &\quad + y_{v+1}^2(\boldsymbol{\alpha}^T\boldsymbol{G}\boldsymbol{\alpha}) - \rho y_{v+1}^2(\boldsymbol{\alpha}^T\boldsymbol{\gamma})^2 \\ &\quad + 2y_{v+1}(\boldsymbol{\beta}^T\boldsymbol{G}\boldsymbol{\alpha}) \\ &\quad - 2\rho y_{v+1}\boldsymbol{\beta}^T\boldsymbol{\gamma}\boldsymbol{\gamma}^T\boldsymbol{\alpha}] \quad (24)\\ &= z^T z + y_{v+1}^2 - [\boldsymbol{\beta}^T\boldsymbol{G}\boldsymbol{\beta} + y_{v+1}^2 \\ &\quad - \rho(y_{v+1} - \boldsymbol{\beta}^T\boldsymbol{\gamma})^2] \\ &= z^T z - \boldsymbol{\beta}^T\boldsymbol{G}\boldsymbol{\beta} + \rho(y_{v+1} - \boldsymbol{\beta}^T\boldsymbol{\gamma})^2 \\ &= \varepsilon(u, v) + \rho(y_{v+1} - \boldsymbol{\beta}^T\boldsymbol{\gamma})^2 \quad (25)\\ &= \varepsilon(u, v) + \rho(y_{v+1} - \boldsymbol{c}^T\boldsymbol{\alpha})^2. \quad (26)\end{aligned}$$

Note that in Eq. (23), Eqs. (21) and (22) were used, and in Eq. (24), the following equation (deduced from Eq. (18)) was used

$$\boldsymbol{\alpha}^T\boldsymbol{\gamma} = \boldsymbol{\alpha}^T\boldsymbol{G}\boldsymbol{\alpha} = \frac{1-\rho}{\rho},$$

which indicates that $0 < \rho < 1$. Up to several manipulations, we also have

$$\rho = \prod_{p=0}^{P} \frac{l-p}{l+p+1}.$$

APPENDIX B
THE HAUSDORFF DISTANCE BETWEEN TWO SETS OF BREAKPOINTS

A sparse vector $\boldsymbol{x} \in \mathbb{R}^N$ is used to represent a set of breakpoints in a signal, and a nonzero entry x_i indicates a breakpoint at the i-th locus with amplitude x_i. Herein, the Hausdorff distance between two sets of breakpoints $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as:

$$d_H(\boldsymbol{x}, \boldsymbol{y}) = \max\{d(\boldsymbol{x}, \boldsymbol{y}), d(\boldsymbol{y}, \boldsymbol{x})\} \quad (27)$$

where $d(\boldsymbol{x}, \boldsymbol{y})$ is the directional Hausdorff distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \max_i \min_j \sqrt{(x_i - y_j)^2 + \nu^2(i-j)^2} \quad (28)$$

ν is a parameter that measures the relative weight between breakpoints' amplitudes and distances. A small value of ν encourages the significance of the amplitude, and *vice-versa*. Generally, we propose to set ν as $\frac{A}{N}$, where A and N are the average absolute value of the breakpoint amplitude and the length of the signal, respectively.

The function *Hausdorff_Dist* by Zachary Danziger is used to calculate d_H, and it is available at https://ww2.mathwor-ks.

TABLE 6. The comparison of different distances. $\nu = 0.2$ for Hausdorff distance.

Distances to $\boldsymbol{x}_1 = [0,1,0,0,0]$	ℓ_2	ℓ_1	ℓ_0	Hausdorff
$\boldsymbol{x}_2 = [0,0,1,0,0]$	1.41	2.00	2.00	0.20
$\boldsymbol{x}_3 = [0,0,1.1,0,0]$	1.49	2.10	2.00	0.22
$\boldsymbol{x}_4 = [0,0,0,0.9,0]$	1.35	1.90	2.00	0.41
$\boldsymbol{x}_5 = [0,0.5,0.5,0,0]$	0.71	1.00	2.00	0.50

cn/matlabcentral/fileexchange/26738-hausdorff-distance?requestedDomain=zh.

In Tab. 6 a toy example is used to show the motivation behind the usage of the Hausdorff distance instead of other commonly used distances, namely, the ℓ_2-, ℓ_1- and ℓ_0- (pseudo)norms. Consider the following five sets of breakpoints: $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5$ as is shown in Tab. 6. $\boldsymbol{x}_1$ is the ground truth, and $\boldsymbol{x}_2$, $\boldsymbol{x}_3$, $\boldsymbol{x}_4$, and $\boldsymbol{x}_5$ are four breakpoint estimates sharing the same ℓ_0 distance from $\boldsymbol{x}_1$. Among these estimates, $\boldsymbol{x}_2$ has the smallest Hausdorff distance from $\boldsymbol{x}_1$ since this breakpoint is located only one point away from the ground truth. Although the ℓ_1 and ℓ_2 distances from $\boldsymbol{x}_5$ to $\boldsymbol{x}_1$ are the smallest, the Hausdorff distance is the largest. This is a good example that shows the advantage of the Hausdorff distance over other norm distances in terms of encouraging separated discontinuities rather than small adjacent discontinuities. The Hausdorff distances of $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ to $\boldsymbol{x}_1$ are in between because of their amplitude biases.

ACKNOWLEDGMENT

The MATLAB code of HOPS can be downloaded at https://www.mathworks.com/matlabcentral/fileexchange/98504-hops-a-fast-algorithm-for-segmenting-piecewise-polynomials.

REFERENCES

[1] W. R. Lai, M. D. Johnson, R. Kucherlapati, and P. J. Park, "Comparative analysis of algorithms for identifying amplifications and deletions in array CGH data," *Bioinformatics*, vol. 21, no. 19, pp. 3763–3770, Oct. 2005.

[2] J. Duan, J.-G. Zhang, H.-W. Deng, and Y.-P. Wang, "CNV-TV: A robust method to discover copy number variation from short sequencing reads," *BMC Bioinf.*, vol. 14, no. 1, pp. 1–12, Dec. 2013.

[3] N. Cahill, S. Rahmstorf, and A. Parnell, "Change points of global temperature," *Environ. Res. Lett.*, vol. 10, pp. 1–6, Aug. 2015.

[4] H.-J. Butt, B. Cappella, and M. Kappl, "Force measurements with the atomic force microscope: Technique, interpretation and applications," *Surf. Sci. Rep.*, vol. 59, nos. 1–6, pp. 1–152, Oct. 2005.

[5] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Signal Process. Mag.*, vol. 16, no. 6, pp. 22–38, Nov. 1999.

[6] E. Mammen and S. van de Geer, "Locally adaptive regression splines," *Ann. Statist.*, vol. 25, no. 1, pp. 387–413, Feb. 1997.

[7] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "ℓ_1 trend filtering," *SIAM Review*, vol. 51, no. 2, pp. 339–360, 2009.

[8] A. B. Olshen, E. S. Venkatraman, R. Lucito, and M. Wigler, "Circular binary segmentation for the analysis of array-based DNA copy number data," *Biostatistics*, vol. 5, no. 4, pp. 557–572, Oct. 2004.

[9] R. Giryes, M. Elad, and A. M. Bruckstein, "Sparsity based methods for overparameterized variational problems," *SIAM J. Imag. Sci.*, vol. 8, no. 3, pp. 2133–2159, Jan. 2015.

[10] M. Novosadova and P. Rajmic, "Piecewise-polynomial signal segmentation using reweighted convex optimization," in *Proc. 40th Int. Conf. Telecommun. Signal Process. (TSP)*, Barcelona, Spain, Jul. 2017, pp. 769–774.

[11] P. Rajmic, M. Novosadová, and M. Daňková, "Piecewise-polynomial signal segmentation using convex optimization," *Kybernetika*, vol. 53, no. 6, pp. 1131–1149, Jan. 2018.

[12] A. Chambolle and P.-L. Lions, "Image recovery via total variation minimization and related problems," *Numerische Math.*, vol. 76, no. 2, pp. 167–188, Apr. 1997.

[13] R. J. Tibshirani, "Adaptive piecewise polynomial estimation via trend filtering," *Ann. Statist.*, vol. 42, no. 1, pp. 285–323, Feb. 2014.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc. B, Methodol.*, vol. 58, no. 1, pp. 267–288, Jan. 1996.

[15] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, Apr. 2004.

[16] J. Duan, C. Soussen, D. Brie, J. Idier, and Y.-P. Wang, "On LARS/homotopy equivalence conditions for over-determined LASSO," *IEEE Signal Process. Lett.*, vol. 19, no. 12, pp. 894–897, Dec. 2012.

[17] Y. C. Eldar and G. Kutyniok, Eds., *Compressed Sensing: Theory and Applications*. Cambridge, U.K.: Cambridge Univ. Press, May 2012.

[18] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *J. Amer. Stat. Assoc.*, vol. 96, no. 456, pp. 1348–1360, Dec. 2001.

[19] R. J. Hodrick and E. C. Prescott, "Postwar U.S. Business cycles: An empirical investigation," *J. Money, credit, Banking*, vol. 29, pp. 1–16, Feb. 1997.

[20] H. Kuroda, M. Yamagishi, and I. Yamada, "Exploiting sparsity in tight-dimensional spaces for piecewise continuous signal recovery," *IEEE Trans. Signal Process.*, vol. 66, no. 24, pp. 6363–6376, Dec. 2018.

[21] I. Auger and C. Lawrence, "Algorithms for the optimal identification of segment neighborhoods," *Bull. Math. Biol.*, vol. 51, no. 1, pp. 39–54, 1989.

[22] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. San, L. Tan, and T. T. Tsai, "An algorithm for optimal partitioning of data on an interval," *IEEE Signal Process. Lett.*, vol. 12, no. 2, pp. 105–108, Feb. 2005.

[23] R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Commun. ACM*, vol. 4, no. 6, p. 284, Jun. 1961.

[24] S. B. Guthery, "Partition regression," *J. Amer. Stat. Assoc.*, vol. 69, no. 348, pp. 945–947, Dec. 1974.

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

[26] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of change-points with a linear computational cost," *J. Amer. Stat. Assoc.*, vol. 107, no. 500, pp. 1590–1598, Dec. 2012.

[27] A. De Cesare and R. Zeboudj, "Algorithmes rapides de restauration des signaux avec prise en compte des discontinuités," in *Proc. Actes 16 Coll. GRETSI*, gretsi, Grenoble, France, Sep. 1997, pp. 1347–1350.

[28] J. Duan, C. Soussen, D. Brie, J. Idier, Y.-P. Wang, and M. Wan, "A parallelizable framework for segmenting piecewise signals," *IEEE Access*, vol. 7, pp. 13217–13229, 2019.

[29] L. R. Turner, "Inverse of the Vandermonde matrix with applications," Nat. Aeronaut. Space Admin., Washington, DC, USA, Nasa Tech. Note, 1966, pp. 1–17.

[30] G. H. Golub, P. Milanfar, and J. Varah, "A stable numerical method for inverting shape from moments," *SIAM J. Sci. Comput.*, vol. 21, no. 4, pp. 1222–1243, Dec. 1999.

[31] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[32] Z. Harchaoui and C. Lévy-Leduc, "Catching change-points with lasso," in *Proc. NIPS*, 2007, pp. 617–624.

[33] J. Duan, C. Soussen, D. Brie, J. Idier, M. Wan, and Y.-P. Wang, "Generalized LASSO with under-determined regularization matrices," *Signal Process.*, vol. 127, pp. 239–246, Oct. 2016.

[34] C. Soussen, J. Idier, D. Brie, and J. Duan, "From Bernoulli–Gaussian deconvolution to sparse signal restoration," *IEEE Trans. Signal Process.*, vol. 59, no. 10, pp. 4572–4584, Oct. 2011.

[35] P. Polyakov, C. Soussen, J. Duan, J. F. L. Duval, D. Brie, and G. Francius, "Automated force volume image processing for biological samples," *PLoS ONE*, vol. 6, no. 4, Apr. 2011, Art. no. e18887.

[36] N. Galvanetto, A. Perissinotto, A. Pedroni, and V. Torre, "Fodis: Software for protein unfolding analysis," *Biophysical J.*, vol. 114, no. 6, pp. 1264–1266, Mar. 2018.

[37] S. Maity, M. Mazzolini, M. Arcangeletti, A. Valbuena, P. Fabris, M. Lazzarino, and V. Torre, "Conformational rearrangements in the transmembrane domain of CNGA1 channels revealed by single-molecule force spectroscopy," *Nature Commun.*, vol. 6, no. 1, pp. 1–15, Nov. 2015.

[38] J. Duan, "Restoration and separation of piecewise polynomial signal. Application to atomic force microscopy," Ph.D. dissertation, Centre de Recherche en Automatique de Nancy, Université Henri Poincaré, Nancy, France, Nov. 2010.