Signal Processing 250 (2027) 110801

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

SplineOP: A dynamic programming knot selection algorithm for time-series compression with quadratic splines

Nicolás Enrique Cecchi [a,*], Vincent Runge [b], Charles Truong [b], Laurent Oudre [a]

[a] Université Paris Saclay, Université Paris Cité, ENS Paris Saclay, CNRS, SSA, INSERM, Centre Borelli, 4 Av des Sciences, Gif-sur-Yvette, F-91190, France
[b] Université Paris Saclay, Université d'Evry Val d'Essonne, CNRS, LaMME, Evry, F-91037, France

ARTICLE INFO

Keywords:
Signal compression
Free-knot splines
Dynamic programming
Change-point detection

ABSTRACT

As the amount of data being processed continues to grow, the need for efficient data representation becomes increasingly important. In this paper, we present `SplineOP`, an algorithm that compresses smooth signals by representing them in the form of quadratic splines. At its core, `SplineOP` aims at solving an L0-penalized spline problem, where both the number and positions of knots are unknown, incorporating ideas from a change-point detection perspective. The problem is expressed through a recurrence structure, which allows for an efficient dynamic programming solution. The algorithm was implemented in C++, with bindings to Python, and tested on synthetic and real datasets. We demonstrate that `SplineOP` successfully recovers the true segment partition as the sampling frequency increases. Moreover, it achieves high-fidelity reconstructions at substantial compression rates for time-series data originating from motion sensors.

1. Introduction

Time-series compression is a necessary part of the data processing pipeline in numerous applications, such as medicine [1,2], energy consumption management [3], IoT [4,5], and manufacturing [6]. Time-series compression consists of finding a low-memory representation of time series.

For *smooth signals*, one popular and classical approach is polynomial splines [7–9]. Polynomial splines can approximate smooth signals with only a few parameters. The approximation is a smooth piecewise polynomial function; the time positions at which the polynomials connect are called *knots*. The compression rate depends on the number of knots used for the approximation, which may vary from signal to another. Fewer knots mean fewer parameters, but the approximation is less accurate. Note that in the context of smooth signals, the approximation has smooth transitions between successive polynomials, guaranteeing continuity of derivatives at the knot positions up to a user-specified degree [10].

Solving the problem with fixed knot positions has been largely studied [11–13]. However, a data-driven choice of the knot positions greatly improves the quality of the compression [15,16]. Unfortunately, for a given signal, finding the knot placement that minimizes the reconstruction error is NP-hard [14]. Several authors have proposed heuristics over the years. Some methods consist of sequentially adding knots where the error is largest [17]. Others place knots at positions of interest, such as local extrema [18]. A well-known approach, called smoothing splines, minimizes a trade-off between the reconstruction error and the L2 norm of the derivative at a user-defined order [12,19]. All of these methods produce close approximations with possibly many unnecessary knots, which are incompatible with compression. Recently, authors have replaced the L2 norm of smoothing splines with the L1 norm. The method, called trend filtering, is a particular case of Lasso regression. Trend filtering yields approximations with fewer knots compared to smoothing splines. Nevertheless, there are still many unwanted knots due to the so-called staircase effect of Lasso [20].

Surprisingly, the use of L0 norm penalization has only been studied in one paper [21]. Compared to L2 and L1 norms, the L0 norm, which is simply the number of nonzero coefficients in a vector, imposes greater sparsity (fewer knots). However, the resulting optimization problem is considerably more difficult to solve.

There exist also modern deep-learning based autoencoder techniques [9,22,23], which achieve high fidelity reconstructions at varying degrees of compression. At the same time, there is no known explicit interpretation of these methods' compressed representations and its size is fixed once the architecture is set, not varying with the complexity of the signal.

In this article, we propose to solve the spline approximation problem using a change-point detection approach. Change-point detection

* Corresponding author.
E-mail addresses: nicolas.cecchi1@ens-paris-saclay.fr (N.E. Cecchi), vincent.runge@univ-evry.fr (V. Runge), ctruong@aup.edu (C. Truong), laurent.oudre@ens-paris-saclay.fr (L. Oudre).

https://doi.org/10.1016/j.sigpro.2026.110801
Received 18 December 2025; Received in revised form 22 May 2026; Accepted 26 June 2026
Available online 27 June 2026

involves estimating the moments when certain characteristics of the data generation process change. One recent advancement in change-point detection is the use of dynamic programming to infer complex data structures by imposing constraints on the sequence of segments. These constraints can be directional (e.g., enforcing up-down changes) or based on graph structures [24,25]. They can also enforce continuity in piecewise linear fits, as in CPOP [26] and slopeOP [27]. Our idea is to leverage these recent advances to propose a new solution for solving piecewise quadratic spline approximation with continuity constraints. These constraints ensure the continuity of the function and its derivative at each knot, allowing a smooth approximation of the signal. The main challenge in achieving this goal is to extend the dynamic programming framework for constrained segmentation beyond piecewise linear approximations.

Our contribution is an algorithm called `SplineOP`, for Spline Optimal Partitioning. `SplineOP` is based on dynamic programming procedure and detects changes in the second-order derivative of quadratic splines by minimizing the squared error regularized with an L0 penalty. To overcome the NP-hardness of the problem, we restrict the set of possible values for our approximation and its derivative at the knots. A solution can then be computed in quadratic time (in the number of samples). We validate its quality on simulated and real data.

The rest of the paper is organized as follows. Section 2 presents the quadratic spline model and its associated L0-regularized optimization problem. In Section 3, we present the dynamic programming solution and its approximation through state and speed discretization. An extensive simulation study in Section 4 shows that `SplineOP` outperforms the alternative methods and can be calibrated with only a small number of labeled samples. An application of the compression of motion sensor data in Section 5 highlights the utility of this method. Short proofs are included in the manuscript, while long ones are sketched and can be found in the Appendices.

2. Mathematical model

In this section, we introduce the quadratic spline model and the associated optimization problem for signal compression.

2.1. Definition and properties of quadratic splines

Definition 1. A **quadratic spline** is a continuous function with a continuous first derivative, defined piecewise by quadratic polynomials. In other words, a quadratic spline f with K knots is a function such that

- f is continuously differentiable;
- There exist K knots τ_k and $K+1$ triples (a_k, b_k, c_k) such that for $\tau \in [\tau_k, \tau_{k+1})$:

$$f(\tau) = a_k(\tau - \tau_k)^2 + b_k(\tau - \tau_k) + c_k, \quad k = 0, \dots, K,$$

where by convention, $\tau_0 = 0$ and $\tau_{K+1} = T-1$, and the knot positions $\boldsymbol{\tau} = \{\tau_k\}_{k=0}^{K+1}$ are defined such that $0 = \tau_0 < \tau_1 < \dots < \tau_{K+1} = T-1$. In a time-series perspective, the values τ_k are integers.

The set of quadratic splines will be denoted $\mathcal{S}_2$ in this article.

Property 1. *Let p_s, p_t, v_t be three real numbers. There exists a unique quadratic function $f(x) = a(x-s)^2 + b(x-s) + c$ such that $f(s) = p_s$, $f(t) = p_t$ and $f'(t) = v_t$.*

Due to the continuity constraints at the knot positions, a quadratic spline exhibits several properties.

Property 2. *Let f be a quadratic spline with knots $\boldsymbol{\tau}$. The parameters $(a_{k-1}, b_{k-1}, c_{k-1})$ and (a_k, b_k, c_k) of two consecutive quadratic polynomials verify the following relationships:*

$$c_k = a_{k-1}(\tau_k - \tau_{k-1})^2 + b_{k-1}(\tau_k - \tau_{k-1}) + c_{k-1},$$
$$b_k = 2a_{k-1}(\tau_k - \tau_{k-1}) + b_{k-1},$$

Property 3. *A quadratic spline f with K knots is fully determined by the knot positions $\{\tau_k\}_{k=1}^{K}$, the values of the function f at the knot positions $\{p_k\}_{k=0}^{K+1}$ (where we define $p_k \triangleq f(\tau_k)$) and the derivative value of the function f at the last knot $v_{K+1} \triangleq f'(\tau_{K+1})$*[1]

Proof. Derivative values at other knots do not need to be explicitly known. Indeed, given that f is piecewise quadratic and that the last speed is v_{K+1}, the only possible value for $f'(\tau_K)$ is $v_K = 2\frac{p_{K+1}-p_K}{\tau_{K+1}-\tau_K} - v_{K+1}$ (using Property 2). By induction, $f'(\tau_k)$ is $v_k = 2\frac{p_{k+1}-p_k}{\tau_{k+1}-\tau_k} - v_{k+1}$. We refer to these relations as the backward propagation of speed. □

In what follows, this set of parameters (knot positions, values at knots, and derivative at last knot) will be denoted by $\mathbf{f}$ and the set of all possible values of the parameters for a quadratic spline with K knots by $\mathcal{F}_K$. In general, $\mathcal{F}_K = [\![1, T-1]\!]^K \times \mathbb{R}^{K+3}$. Note that knowing $\mathbf{f} \in \mathcal{F}_K$ is equivalent to knowing f.

2.2. Model and problem formulation

In this article, we consider observations $\{y_t\}_{t=0}^{T-1} \in \mathbb{R}^T$ sampled at successive times $t = 0, \dots, T-1$. We assume that y_t is modeled as

$$y_t = f(t) + \epsilon_t, \quad t = 0, \dots, T-1, \tag{1}$$

where $f : [0, +\infty) \to \mathbb{R}$ is a quadratic spline and $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ are independent and identically distributed (iid) Gaussian variables with variance σ^2.

Our objective is to estimate the positions of the knots from the samples $\{y_t\}_{t=0}^{T-1}$. To do this, we will use a change-point detection approach to detect the positions $\tau_1, \dots, \tau_K$ where the jumps in the second derivative occur, i.e., $f''(\tau_k^-) \neq f''(\tau_k^+)$, while taking into account continuity constraints. Note that along the manuscript, we will use the terms change point and knot interchangeably.

A common approach to change-point detection involves minimizing the penalized reconstruction error:

$$\min_{f \in \mathcal{S}_2} \left\{ \sum_{t=0}^{T-1} \|y_t - f(t)\|^2 + \beta(f, y) \right\}, \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm, and $\beta(\cdot,\cdot)$ is a data-dependent regularization function that penalizes over-complex splines. A simple penalty function is $\beta(f, y) \propto K(f)$ [28] where $K(f)$ is the number of knots of the spline f. This penalty is called "linear" and favors splines with few knots. Relation (2) becomes:

$$\min_{f \in \mathcal{S}_2} \left\{ \sum_{t=0}^{T-1} \|y_t - f(t)\|^2 + \beta K(f) \right\}, \tag{3}$$

where $\beta > 0$ is a user-defined penalty value that controls the trade-off between data fidelity and spline complexity.

Remark 1. Without the continuity constraints, the optimization problem (3) can be solved with standard dynamic programming techniques with a complexity of $\mathcal{O}(T^2)$. However, continuity constraints at knot positions make the problem much more difficult to solve and prevent the use of these standard techniques [26,27].

Remark 2. We consider the regularly sampled setting for simplicity, as the extension to irregularly sampled time series is straightforward. In appendices, proofs are given for this more general case.

Remark 3. Our method naturally extends to functions taking values in $\mathbb{R}^d$, viewed as d independent instances of the previously defined f, each with its own quadratic coefficients, but with identical change positions. For clarity, we do not emphasize the multivariate structure, since all forthcoming notations can be interpreted either in $\mathbb{R}$ or in $\mathbb{R}^d$.

[1] or any other position.

3. Fitting quadratic splines with dynamic programming

To solve the optimization problem (3), we propose to efficiently explore the space of quadratic splines. In Section 3.1 we first introduce the optimal solution of the problem, which is computationally intractable. To overcome this issue, we introduce our algorithm, `SplineOP` (Spline Optimal Partitioning), in Section 3.2, which proposes to compute an approximate solution to the problem by relying on several simplification strategies:

1. (for positions) discretizing the values the spline can take at the knots;
2. (for initial speed) discretizing the values of the first derivative at the leftmost point;
3. (for speeds) propagating first derivative information forward.

3.1. Construction of the optimal dynamic programming solution

In the following, a segment is defined as integer interval $s : t := \{s, \dots, t-1\}$. The reconstruction error, also called cost, on the segment $s : t$ is defined by:

$$C_{s:t}(p_s, p_t, v_t) := \sum_{i=s}^{t-1} \|f(i) - y_i\|^2 . \tag{4}$$

As seen in Property 1, it only depends on the parameters p_s, p_t and v_t. Thanks to Property 3, we can rewrite the minimization (3) over $\mathcal{S}_2$ – the space of continuous piecewise quadratic splines – as a minimization over the set of (i) all possible numbers of changes K, (ii) knot positions, (iii) values of the functions at the knots and (iv) final speed, with the constraints that the speeds v_k satisfy the previous recursive relations. By denoting $\mathbf{f} \in \mathcal{F}_K$ the set of parameters needed to fully characterize function f, we can formulate the optimization problem as

$$\begin{cases} \min\limits_{K, \mathbf{f} \in \mathcal{F}_K} \sum\limits_{k=1}^{K+1} \left[C_{\tau_{k-1}:\tau_k}(p_{k-1}, p_k, v_k) + \beta \right], \\ \text{s.t. } \; v_{k-1} = 2\,\dfrac{p_k - p_{k-1}}{\tau_k - \tau_{k-1}} - v_k, \quad \forall k \in \{1, \dots, K+1\} . \end{cases} \tag{5}$$

We present a dynamic programming approach to find the minimizer of (5). Let $Q^{(t)}(p, v)$ be the optimal solution of (5) when approximating the observations up to (but not including) time t, $\{y_i\}_{i=0}^{t-1}$, with a quadratic spline, with final position p and final speed v occurring at t. We get the following result:

Proposition 1. *The following recurrence relation holds true:*

$$Q^{(t)}(p, v) = \min_{\substack{\tilde{p} \in \mathbb{R} \\ 0 \le s < t}} \left\{ Q^{(s)}\left(\tilde{p}, 2\frac{p - \tilde{p}}{t - s} - v\right) + C_{s:t}(\tilde{p}, p, v) + \beta \right\}, \tag{6}$$

with $Q^{(0)}(\cdot, \cdot) = 0$.

Notice that the proof of this proposition in Appendix B is based on the search for the best last changepoint position, s, when t points have been observed. The continuity constraints at s impose an additional optimization over $f(s) = \tilde{p}$ and the backward propagation of the final speed to ensure speed continuity. The piecewise function $Q^{(t)}(\cdot, \cdot)$ becomes increasingly complex as t grows; we can get up to 2^{t-1} quadratics in parameters (p, v) in $Q^{(t)}(\cdot, \cdot)$. Currently, there is no efficient method to solve this problem in a limited time. As stated in Remark 1, this intractability is only due to the continuity constraints. For unconstrained algorithms, in contrast, the number of possible solutions in $Q^{(t)}$ grows linearly with t, reaching at most t pieces, as shown in [29]. This illustrates the intrinsic difficulty of our problem; as a result, solving the recurrence in Proposition 1 remains intractable.

Equivalently, we can present the recursion in a forward speed description, defining the function $\vec{Q}^{(t)}(p, v_0)$ as the optimal solution of (5) with final position p, but initial speed v_0.

Remark 4. In a forward description, the obtained recursion is as follows:

$$\vec{Q}^{(t)}(p, v_0) = \min_{\substack{\tilde{p} \in \mathbb{R} \\ 0 \le s < t}} \left\{ \vec{Q}^{(s)}\left(\tilde{p}, v_0\right) + C_{s:t}(\tilde{p}, p, g_{s:t}(\tilde{p}, v_0)) + \beta \right\}, \tag{7}$$

with $\vec{Q}^{(0)}(\cdot, \cdot) = 0$. Function $g_{s:t}$ takes the form $2\sum_{i=1}^{k}(-1)^{k-i}\frac{p_i - p_{i-1}}{\tau_i - \tau_{i-1}} + (-1)^k v_0$ specific to the value $\tilde{p}$ $(= p_k)$ and v_0 with $\tau_{k-1} = s$ and $\tau_k = t$. This is the forward speed propagation formula of the optimal path ending at value $\tilde{p}$ with initial speed v_0.

Values $\vec{Q}^{(t)}(p, v_0)$ and $Q^{(t)}(p, v)$ coincide if the initial speed v_0 leads to the final speed v (using forward propagation). The complexity of this problem remains the same. To make the problem tractable, we will reduce the choice of the final speed to one value for each position p; we write:

$$\bar{Q}^{(t)}(p, v_t(p)) = \min_{\substack{\tilde{p} \in \mathbb{R} \\ 0 \le s < t}} \left\{ \bar{Q}^{(s)}\left(\tilde{p}, v_s(\tilde{p})\right) + C_{s:t}(\tilde{p}, p, 2\frac{p - \tilde{p}}{t - s} - v_s(\tilde{p})) + \beta \right\}, \tag{8}$$

with $v_t(p)$ the value $2\frac{p - \tilde{p}}{t - s} - v_s(\tilde{p})$ selected by the arguments of the minimum. Then, the `SplineOP` algorithm solves this recursion for a small set of values for p and v_0. This strategy is fully detailed in the next section.

3.2. The SplineOP algorithm

In (5), the optimal parameters $\mathbf{f}$ are supposed to belong to the set $\mathcal{F}_K$, where positions and speed are continuous parameters. To solve the problem, we propose discretizing these parameters to form a finite discrete set, and we utilize the recursion (8) based on the forward propagation of speeds.

To do so, we will define the following sets:

- Each function value $f(t)$ (for integer t) will be assumed to belong to a finite set of candidate values $\mathcal{P}_t = \{p_{t,1}, p_{t,2}, \dots, p_{t,n}\}$. For simplicity, all these sets are of size n;
- The initial derivative of the spline $f'(0)$ will be assumed to belong to a finite set of candidate values $\mathcal{V}_0 = \{v_{0,1}, v_{0,2}, \dots v_{0,m}\}$ of size m.

This discretization is equivalent to switching from a possible set of parameters $\mathcal{F}_K = [\![1, T-1]\!]^K \times \mathbb{R}^{K+3}$ to a set $\tilde{\mathcal{F}}_K = [\![1, T-1]\!]^K \times \{\mathcal{P}_{\tau_k}\}_{k=0}^{K+1} \times \mathcal{V}_0$[2]. We later describe heuristics for choosing sets $\mathcal{V}_0$ and $\mathcal{P}_t$.

Note that, thanks to this discretization, the set of possible speeds $\mathcal{V}_t$ at time t is also discrete.

The speed $v_{t,j}$ at time t and position $p_{t,j}$ is the propagation of the speed $v_{s,i}$ at the beginning of the last segment of the optimal solution, starting at time s and position $p_{s,i}$. Let $\tilde{Q}_j^{(t)} = \bar{Q}^{(t)}(p_{t,j}, v_{t,j})$ be the best cost of fitting observations $\{y_s\}_{s=0}^{t-1}$ with a spline $\tilde{f}$ such that $\tilde{f}(t) = p_{t,j}$. This best cost $\tilde{Q}_j^{(t)}$ is the minimum between two alternatives:

$$\tilde{Q}_j^{(t)} = \min\{Q_j^{(t),\,no\,change}, Q_j^{(t),\,change}\}, \tag{9}$$

where

$$Q_j^{(t),\,no\,change} = \min_{\substack{v_0 \in \mathcal{V}_0 \\ 1 \le i \le n}} \left\{ C_{0:t}(p_{0,i}, p_{t,j}, v_{t,j}^{cand}(v_0, i)), \quad v_{t,j}^{cand}(v_0, i) = 2\frac{p_{t,j} - p_{0,i}}{t} - v_0 \right\}, \tag{10}$$

$$Q_j^{(t),\,change} = \min_{\substack{0 < s < t \\ 1 \le i \le n}} \left\{ Q_i^{(s)} + C_{s:t}(p_{s,i}, p_{t,j}, v_{t,j}^{cand}(s, i)) + \beta, \quad v_{t,j}^{cand}(s, i) = 2\frac{p_{t,j} - p_{s,i}}{t - s} - v_{s,i} \right\}. \tag{11}$$

The final speed $v_{t,j}$ is the candidate $v_{t,j}^{cand}$ computed at the optimal indices (v_0, i) or (s, i). This value associated with the optimal cost is stored

[2] with a little abuse of notation, as the τ_k values are first chosen in the set $[\![1, T-1]\!]^K$.

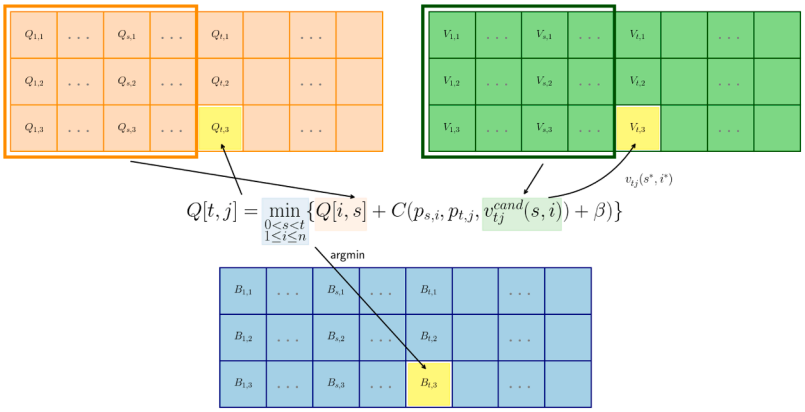


Fig. 1. Relationship between the optimization step and matrices Q (orange), V (green), and B (blue). Yellow boxes show the values being filled for the current (time, state) pair, (t, j). Bounding boxes in Q and V show the values previously computed, which are needed to compute the current solution. For times greater than the current t, the matrices are empty.

to be used at future time-steps. To reconstruct the final vector of change-points, we need to store the duple of best previous time and state, $(s^\star, i^\star)$, for each final time-state (t, j) when solving problems (10) and (11).

The resulting iterative algorithm is called `SplineOP`[3] (Spline Optimal Partitioning); its pseudo-code is given in Algorithm 1. The algorithm is based on the recursive computations of

- $\mathbf{Q} \in \mathbb{R}^{(T+1)\times n}$ that stores the best cost $\tilde{Q}_j^{(t)}$ of fitting the signal $\{y_s\}_{s=0}^{t-1}$ with a quadratic spline $\tilde{f}$ with $\tilde{f}(t) = p_{t,j}$;
- $\mathbf{B} \in (\mathbb{N}^2)^{(T+1)\times n}$ that stores the duples $(s^\star, i^\star)$ that were used to optimize (10) or (11) for each t and j value;
- $\mathbf{V} \in \mathbb{R}^{(T+1)\times n}$ that stores final speeds obtained when optimizing (10) or (11) for each t and j values.

The schema in Fig. 1 shows the relationship between these matrices and the optimization step.

Algorithm 1 `SplineOP` algorithm.

input: observations $\mathbf{y} = \{y_t\}_{t=0}^{T-1}$, penalty β, T sets of n states $\mathcal{P}_t$, initial speeds $\mathcal{V}_0$ with m speeds
$\mathbf{Q}[0, i] = 0, \quad i = 1, \dots, n$ ▷ Initialization
$\mathbf{V}[0, i] = \mathcal{V}_0, \quad i = 1, \dots, m$
for $t = 1$ to T **do** ▷ Loop over ending time
  for $j = 1$ to n **do** ▷ Loop over ending states in $\mathcal{P}_t$
    $\mathbf{Q}[t, j] = \min\left\{Q_{t,j}^{no\,change}, Q_{t,j}^{change}\right\}$
    $\mathbf{B}[t, j] = (s^\star, i^\star)$ ▷ See definition above
    $\mathbf{V}[t, j] = 2\frac{p_{t,j} - p_{s^\star,i^\star}}{t - s^\star} - \mathbf{V}[s^\star, i^\star]$
  end for
end for
$\hat{\boldsymbol{\tau}} \leftarrow$ backtrack_changes($\mathbf{B}, \mathbf{Q}$)
return $\hat{\boldsymbol{\tau}}$

Once the double for-loop is over, the change-point vector is computed by a backtracking algorithm, see Algorithm 2 below. The algorithm first finds the best state index j at the final point $\tau = T - 1$. It then iteratively backtracks through the matrix $\mathbf{B}$, using the matrix content to update the indices (τ, j), thereby reconstructing the optimal solution path. The time indices are stored, constructing a change-point vector $\hat{\boldsymbol{\tau}}$.

Proposition 2. *(Time complexity) `SplineOP`'s time complexity is $\mathcal{O}(n^2T^2)$.*

[3] An implementation is available at https://github.com/nicolascecchi/SplineOP

Algorithm 2 `backtracking` algorithm.

$(\tau, j) = (T, \operatorname{argmin}_{j=1,\dots,n} \mathbf{Q}[T, j])$
$\hat{\boldsymbol{\tau}} \leftarrow (\tau)$ ▷ Initialize vector of changes.
do
  $(\tau, j) = \mathbf{B}[\tau, j]$
  $\hat{\boldsymbol{\tau}} \leftarrow (\tau, \hat{\boldsymbol{\tau}})$ ▷ Prepend previous change point to $\hat{\boldsymbol{\tau}}$.
while $\tau > 0$
return $\hat{\boldsymbol{\tau}}$

Proof. There are three matrices to fill ($\mathbf{Q}$, $\mathbf{B}$, $\mathbf{V}$), each of size $(T + 1) \times n$. At time t, we compare at most tn values in (10) and (11) in $\mathcal{O}(1)$ time (Lemma 1, below). For each matrix, we get a time upper bounded by $\sum_{t=1}^{T+1}\sum_{j=1}^{n}(tn\mathcal{O}(1)) \le \mathcal{O}(n^2T^2)$, which proves the result. □

Lemma 1. *Assume that the following cumulative sums are precomputed for any $t \le T$: $\sum_{s\le t} y_s$, $\sum_{s\le t} y_s^2$, $\sum_{s\le t} s y_s$, and $\sum_{s\le t} s^2 y_s$, which can be done in $\mathcal{O}(T)$ operations. Then, for fixed parameters (s, t, p_s, p_t, v_t), the value of the segment cost $C_{s:t}(p_s, p_t, v_t)$ (4) can be computed in $\mathcal{O}(1)$ operations.*

Proof. The proof in Appendix A is mostly algebraic manipulation. □

Notice that exact pruning strategies cannot be easily incorporated to improve execution time (see [27]), since segment costs are not based on parameter optimization, and standard PELT pruning [28] cannot, therefore, be applied. Some heuristic strategies could be developed to remove indices s to be considered; this is left for further study.

Proposition 3. *(Non-optimality) The algorithm does not guarantee convergence to the global minimizer of (5), due to the greedy (or implicit) pruning of the speed: at time t and position j, only a unique speed is available, $v_{t,j}$.*

Proof. A simple counterexample can be seen in Appendix C. The proof is based on the fact that the optimal solution up to a certain time t generates a single value of the first derivative, which must be respected in the future. This first derivative value may not be optimal for the upcoming data. □

3.3. Discretization of positions and speeds

Our algorithm, `SplineOP`, requires a set of initial speeds $\mathcal{V}_0$ and candidate values $\mathcal{P}_t$ for each timestamp $0 \le t \le T - 1$. One would like to choose these as close as possible to their true values. Below, we discuss the heuristics to generate these sets.

Positions discretization. At each time step one data point is available, which is a noisy observation of the true value. A sensitive strategy to build the sets of candidate values $\mathcal{P}_t$ is by including the observation at

time t and points around it. For our experiments, we have fixed $|\mathcal{P}_t| = 5$ and we have used a randomized strategy, taking the four additional points as random samples from a Gaussian distribution $\mathcal{N}(y_t, \hat{\sigma}^2_{\text{Hall}})$, centered at the observation y_t with variance $\hat{\sigma}^2_{\text{Hall}}$ estimated using the Hall estimator [30]. Other strategies were evaluated and had a worse performance, for details see Appendix D.

Initial speeds discretization. The size of the set of initial speeds, $|\mathcal{V}_0|$, was also set to 5 by default. The values of the estimated speeds are generated by computing the least-squares quadratic polynomial approximation to the first 20, 40, 60, 80 and 100 data points, and taking its first derivative.

3.4. Compression routine

Algorithm 3 proposes a compression routine based on the results of `SplineOP`. First, `SplineOP` is run to detect change points in acceleration. The change points $\hat{\boldsymbol{\tau}}$ are used as knots to fit a least-squares quadratic spline. The resulting spline can be retrieved by storing only a small number of parameters (a_k, b_k, c_k, τ_k) for each segment with $k \in \{1, \dots, K\}$ compared to the signal size (see Appendix F).

Algorithm 3 `SplineOP compression routine.`

input: observations $\mathbf{y} = \{y_t\}_{t=0}^{T-1}$, penalty β, T sets of n states $\mathcal{P}_t$, initial speeds $\mathcal{V}_0$
$\hat{\boldsymbol{\tau}} \leftarrow \text{SplineOP}(\mathbf{y}, \beta, \mathcal{P}, \mathcal{V}_0)$
spline $\leftarrow$ least_squares_fit$(\mathbf{y}, \hat{\boldsymbol{\tau}})$
return spline

4. Synthetic data experiments

In this section, we assess the ability of our method, `SplineOP`, to estimate change-point positions from noisy time series and the ability of our compression routine (Algorithm 3) to reconstruct the true underlying signal.

4.1. Setting

Data. A synthetic dataset is generated by regularly sampling 3D quadratic splines with 4, 9, and 14 knots, positioned randomly and shared across the 3 dimensions. The knot positions follow a Dirichlet distribution. Unless otherwise specified, the length of each signal is 1000 points. For each segment, the acceleration amplitude follows a uniform distribution on the interval $[10, 15]$. Between two consecutive segments, the acceleration sign alternates between $+1$ and -1. The rest of the parameters can be computed from this construction, see Appendix F. Examples of these signals can be seen in Fig. 2. Finally, Gaussian noise is added to each spline, such that SNR = 30 dB, with the following definition:

$$\text{SNR} = 20 \log \frac{\sigma_{signal}}{\sigma_{noise}}.$$

For each number of knots (4, 9 or 14), we generate 100 signals.

Metrics. To evaluate reconstruction quality, we use the mean squared error:

$$MSE = \frac{1}{T} ||y_{obs} - y_{pred}||_2^2, \tag{12}$$

where y_{pred} are the predictions made by the least-squares spline constructed with the knots found by `SplineOP`.

To evaluate the change-point detection ability, the F-score is used. An estimated change point $\hat{t}$ is a true positive if there is a true change point t^* such that $|\hat{t} - t^*| < m$, where m is a user-defined margin. In our experiments, m is 2% of the total length of the signal. Define the F-Score by:

$$\text{F-score} = \frac{2}{\text{precision}^{-1} + \text{recall}^{-1}}, \tag{13}$$

where

$$\text{precision} = \frac{\#\text{ True positives}}{\#\text{ Estimated change points}} \text{ and}$$

$$\text{recall} = \frac{\#\text{ True positives}}{\#\text{ Real change points}}.$$

Table 1
Size of the compressed signal for 3 different approaches. K_m is the total number of change points for sensor m.

Strategy	S_c
All-together	$K_m(MD+1)+3MD$
One-by-one (penalized)	$\sum_{m=1}^{M}[K_m(D+1)]+3D$
One-by-one (constrained)	$M[K_m(D+1)+3D]$

Baselines. We include two baselines: L1 trend-filtering (`L1TF`) [31] and a naive approach `naive`. `L1TF` is a lasso-based approach with one user-defined parameter, the penalty. We use the Bayesian Information Criterion (BIC) to set its value. The naive approach involves smoothing the signal, examining the 3rd-order discrete derivative, and identifying local extrema as potential change points. As shown in [32], the approach proposed in [21], based on an L0 penalty, yields worse compressed approximations than L1 trend filtering; therefore, it is not considered further.

4.2. Results

Reconstruction error. We first analyze the reconstruction error for different numbers of predicted change points $\hat{K}$. From Fig. 3 we make two observations:

- `SplineOP` outperforms the `naive` approach. `SplineOP` also outperforms `L1TF` for most values of $\hat{K}$. When $\hat{K}$ is much larger than the true number of change points K^*, they have similar performance.
- `SplineOP` has a natural behaviour: increasing $\hat{K}$ improves the MSE, up to $\hat{K} = K^*$. Beyond that point, the MSE does not improve. `L1TF` and `naive` do not have the same behaviour.

Influence of sampling frequency on F-Score. We evaluate the effect of sampling frequency for the change-point detection task. We sample the splines of our dataset at varying frequencies. `SplineOP` is given the true number of change points K^* as input. As seen on Fig. 4, higher sampling frequencies imply a higher F-score. Past a certain point, improvement stalls.

Influence of the number of states. We run the algorithm with different numbers of states $|\mathcal{P}_t|$ to see the effect on MSE, Fscore and execution time. Fig. 5 shows that, on average, adding states lowers the reconstruction error, while the Fscore at 2% margin is mostly unchanged.

The computation time grows quadratically, as proved in Proposition 2. The empirical results are shown in Fig. 6

Penalty learning. Certain datasets have labeled signals, meaning that an expert has manually annotated the change-point locations. We show that this information can be used to find a suitable penalty β that maximizes the F-score. To that end, we split the labeled data into a training set and a test set, and run a grid search on the training set to find the

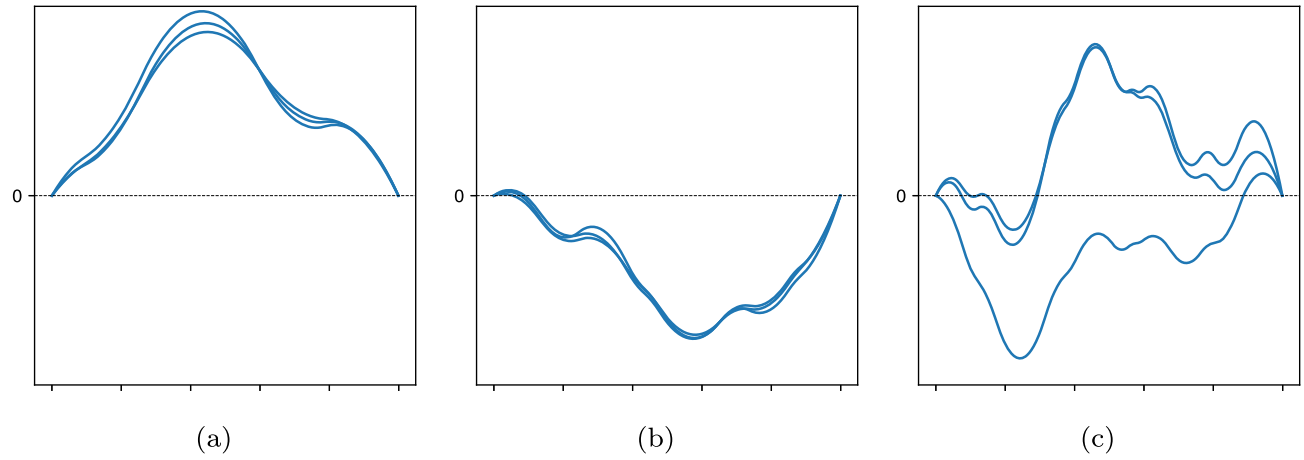


Fig. 2. Examples of synthetic data used in the experiences. 2a 4 change points, 2b 9 change points and 2c 14 change points.

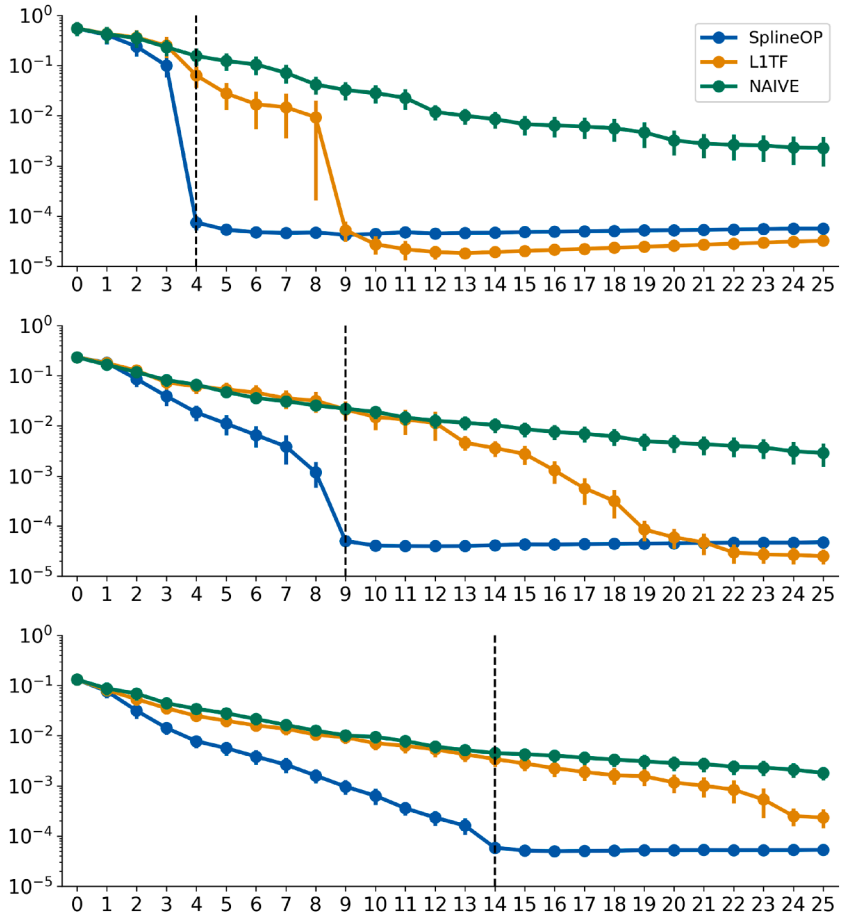


Fig. 3. MSE values for different numbers of change points. Dots show the mean over all signals, and bars show the standard deviation. Dashed black line indicating the real number of change points in the family of signals. Top: 4 change points; middle: 9 change points; bottom: 14 change points.

optimal value of β. Both datasets have equal and balanced proportions of signals with 4, 9, and 14 change points. We consider two train set sizes: 10% and 80% of the dataset. Table 2 shows there is no significant difference in performance by using 10% or 80% of the dataset to learn the penalty β. In other words, a user does not need to label many signals.

5. Real data experiments

5.1. Setting

Data. We use `SplineOP` to compress signals from the ARM-Coda dataset [33]. The dataset is a collection of 34 movement sensors placed on 16

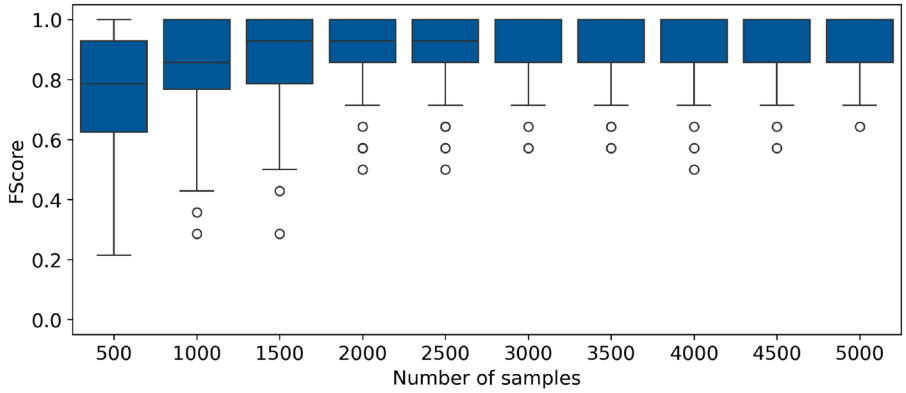


Fig. 4. F-score with a margin of 2% of signal length.

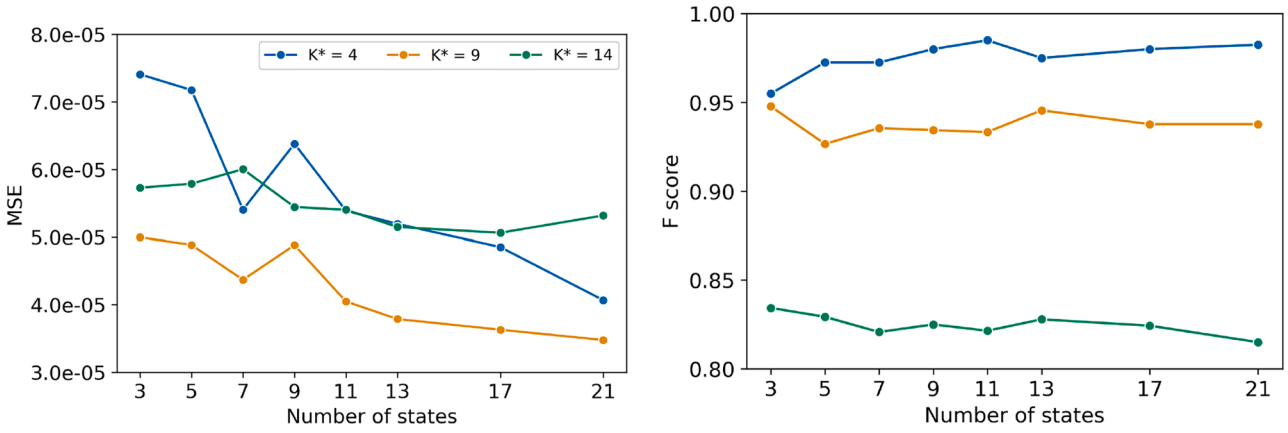


Fig. 5. Impact of the number of states on the performance of the algorithm. Evolution of the MSE (left). Evolution of the F-score with 2% of margin (right).

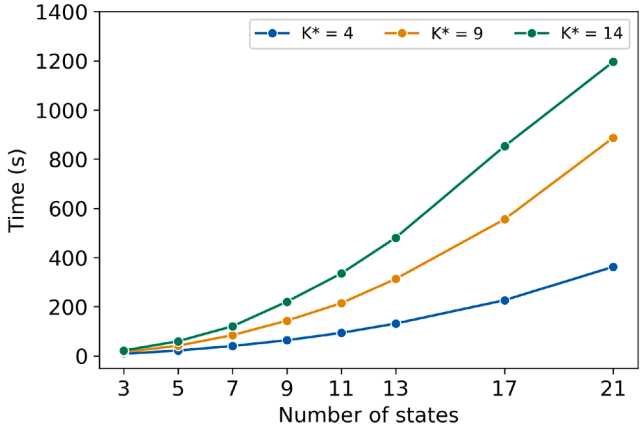


Fig. 6. Time complexity of SplineOP with respect to the number of states. Let $T_K(n)$ be the time complexity as a function of the number of states n for K changepoints, from the simulations we find $T_4(n) \sim n^{1.93}$, $T_9(n) \sim n^{2.09}$ and $T_{14}(n) \sim n^{2.11}$.

healthy individuals that perform 15 different tasks. The sensors record 3D position data at 100 Hz.

Here we focus on `subject 0` doing `movement 5` (combing their hair with the right arm). The time series lasts 40 s (4 000 points).

Metric. We consider the task of time-series compression. The compression rate is defined as:

$$\text{compression rate} = \frac{\text{size of uncompressed signal}}{\text{size of compressed signal}} = \frac{S_u}{S_c}, \tag{14}$$

with

$$S_u = M \cdot D \cdot T = 34 \cdot 3 \cdot 4\,000 = 408\,000$$

where $M = 34$ is the number of sensors; $D = 3$, the dimensions; and $T = 4\,000$, the number of observations. On the other hand, S_c depends on the compression strategy:

- *One-by-one (penalized)* Each sensor is considered individually (there are 34 three-dimensional signals). The penalty value is the same for all. The numbers and positions of the predicted change points differ between sensors.

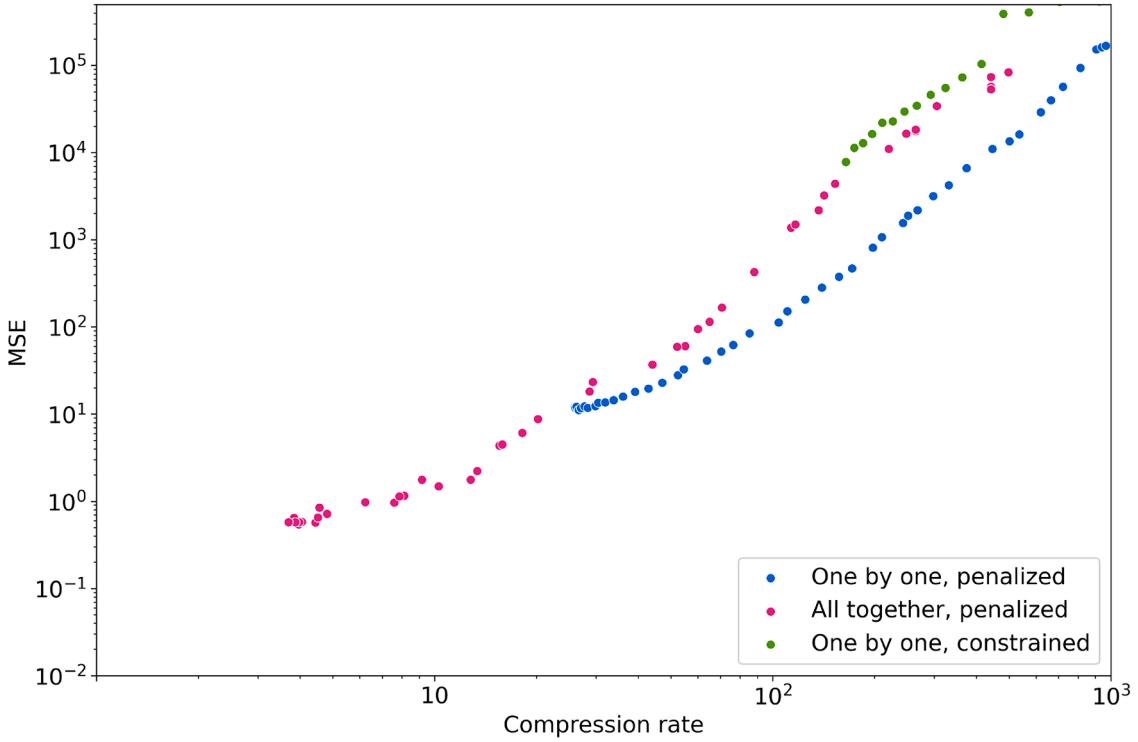
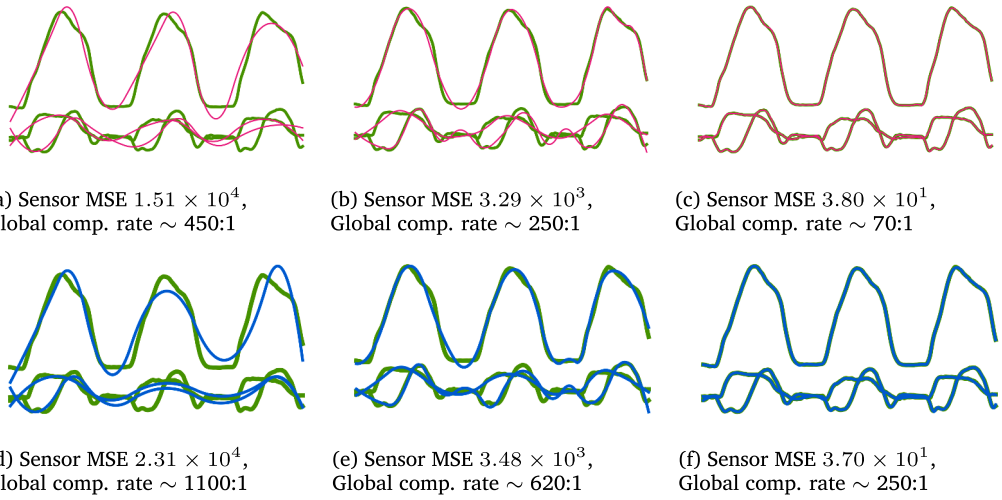


Fig. 7. Overall MSE for different compression rates. *One-by-one (penalized)* strategy performs consistently better than *All-together*.



(a) Sensor MSE 1.51×10^4, Global comp. rate $\sim$ 450:1

(b) Sensor MSE 3.29×10^3, Global comp. rate $\sim$ 250:1

(c) Sensor MSE 3.80×10^1, Global comp. rate $\sim$ 70:1

(d) Sensor MSE 2.31×10^4, Global comp. rate $\sim$ 1100:1

(e) Sensor MSE 3.48×10^3, Global comp. rate $\sim$ 620:1

(f) Sensor MSE 3.70×10^1, Global comp. rate $\sim$ 250:1

Fig. 8. Reconstruction of the three dimensions of the observed (green) signal on a sensor placed on the right arm *(sensor 21)*. Top: *All-together (penalized)* strategy. Bottom: *One-by-one (penalized)* strategy. Each column has reconstruction errors of similar magnitude. The compression rate attained by the strategy at the bottom is preferable.

- *One-by-one (constrained)* Each sensor is considered individually (there are 34 three-dimensional signals). The number of estimated change points is the same for all sensors. The positions of the predicted change points differ between sensors.
- *All-together* All sensors are considered simultaneously (a single 102 dimensional signal). The number of estimated change points is the same for all sensors. The positions of the predicted change points are the same for all sensors.

The size S_c of the compressed signal is given in Table 1 for each strategy. As an example, if `SplineOP` has a compression rate of 200:1, it means that `SplineOP` only need 1 parameter to model 200 points.

5.2. Results

Fig. 7 shows the MSE of these three strategies for different compression rates. *One-by-one (constrained)* has observations for compression

Table 2
F-score (mean and standard deviation) on a test set using the penalties learned with 10% and 80% of the data. The p-value is according to the Mann-Whitney-Wilcoxon test. According to this test, the two F-scores are not significantly different.

Penalty	F-score
$\beta_{10\%}$	0.81 (0.13)
$\beta_{80\%}$	0.79 (0.16)
p-value	0.76

rates above 160:1 only. Smaller compression rates require longer runs for each signal and are skipped.

For a given compression rate, *One-by-one (penalized)* outperforms *All-together*. This is because *All-together* is less flexible; all change points are shared across sensors. *One-by-one (penalized)* can add more change points to sensors that vary a lot and less change points to sensors that do not move much.

A reconstruction example is shown on Fig. 8. Again, we see that, for a given reconstruction error, *One-by-one (penalized)* produces approximations with less parameters.

6. Conclusions

In this paper, we introduce `SplineOP`, a change-point detection algorithm that can efficiently place knots to fit a quadratic spline to data. The algorithm outperforms baselines on both synthetic and real datasets. We also show that the penalty value can be calibrated using only a small number of samples without significant loss in performance.

For future work, adapting this algorithm to higher-noise settings is challenging, as it requires a more careful selection of state values for discretized positions. The potential loss of optimality due to forward speed propagation must be thoroughly studied to better understand the approximation quality of our method. It may be possible to design an algorithm with continuous initial speed values, drawing inspiration from functional pruning techniques developed in the multiple change-point literature. A heuristic approach for pruning positions using a PELT-like rule is also a potential direction of improvement for speeding up the algorithm.

CRediT authorship contribution statement

Nicolás Enrique Cecchi: Writing – review & editing, Writing – original draft, Visualization, Investigation, Formal analysis, Conceptualization; **Vincent Runge:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization; **Charles Truong:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization; **Laurent Oudre:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been partially funded by the Industrial Data Analytics and Machine Learning chair of ENS Paris-Saclay.

Data availability

Code available on a public repository. Data available on request.

Appendix A. Proof of Lemma 1

Given an interval $[x_s, x_t)$ and values (p_s, p_t, v_t), a quadratic polynomial p satisfying $p(x_s) = p_s$, $p(x_t) = p_t$, and $p'(x_t) = v_t$ is uniquely defined and can be written as $p(x) = a(x - x_s)^2 + b(x - x_s) + c$ for $x \in [x_s, x_t)$, with appropriate a, b, c. Define the length of the interval from the beginning of the segment as: $L_i = x_i - x_s$. We develop the square in the cost formula:

$$\begin{aligned} C_{s:t}(p_s, p_t, v_t) &= \sum_{i=s}^{t-1} (y_i - p(x_i))^2 = \sum_{i=s}^{t-1} (y_i - aL_i^2 - bL_i - c)^2 \\ &= \sum_{i=s}^{t-1} (y_i^2 - 2y_i(aL_i^2 + bL_i + c) + (aL_i^2 + bL_i + c)^2) \\ &= a^2[L_i^4]_s^t + 2ab[L_i^3]_s^t - 2a[y_i L_i^2]_s^t + [y_i^2]_s^t - 2b[y_i L_i]_s^t \\ &\qquad + (2ac + b^2)[L_i^2]_s^t - 2c[y_i]_s^t + 2cb[L_i]_s^t + [c^2]_s^t, \end{aligned}$$

where $[\cdot]_s^t := \sum_{i=s}^{t-1} \cdot = cumsum_{t-1}(\cdot) - cumsum_{s-1}(\cdot)$. If the cumulative sums are precomputed, this operation is reduced to retrieving the corresponding values, which is $\mathcal{O}(1)$. In particular:

$$\sum_{i=s}^{t-1} L_i y_i = \sum_{i=s}^{t-1} (x_i - x_s) y_i = [x_i y_i]_s^t - x_s [y_i]_s^t,$$

and,

$$\sum_{i=s}^{t-1} L_i^2 y_i = \sum_{i=s}^{t-1} (x_i - x_s)^2 y_i = [x_i^2 y_i]_s^t - 2x_s [x_i y_i]_s^t + x_s^2 [y_i]_s^t .$$

If the sampling is not uniform, then one needs to develop the binomial expansion:

$$\sum_{i=s}^{t-1} L_i^d = \sum_{i=s}^{t-1} (x_i - x_s)^d = \sum_{j=0}^{d} \binom{d}{j} (-x_s)^j \Big(\sum_{i=s}^{t-1} x_i^{d-j} \Big) = \sum_{j=0}^{d} \binom{d}{j} (-x_s)^j [x_i^{d-j}]_s^t .$$

Consequently, we can compute such sums using the cumulative sums for the x_i^{d-j} $(x_i, x_i^2, \dots, x_i^d)$ and we get the same $O(1)$ time and then the same result.

This can be simplified if the sampling is equispaced. Let δ be the spacing between samples, then:

$$\sum_{i=s}^{t-1} L_i^d = \sum_{i=s}^{t-1} (\delta(i - s))^d = \delta^d \sum_{k=0}^{t-s-1} k^d = \delta^d [k^d]_1^{t-s},$$

which is the sum of the first $t - s - 1$ integers to the power d and can be computed with Faulhaber's formula. Notice that, for multivariate time series, we sum analogous cost terms – differing only in their respective parameter values a, b, c – across all dimensions. This yields the same result in complexity.

Appendix B. Proof of Proposition 1

The set $\mathcal{T}_t$ denotes all partition vectors $\boldsymbol{\tau} = \{\tau_k\}_{k=0}^{K+1}$ satisfying $0 = \tau_0 < \tau_1 < \cdots < \tau_K < \tau_{K+1} = t$. The number of interior points K is not fixed, and all possible partition lengths are included in $\mathcal{T}_t$. Consider the problem in Eq. 5 given by:

$$Q^{(t)} = \min_{\substack{\boldsymbol{\tau} \in \mathcal{T}_t \\ p \in \mathbb{R}^{K+2} \\ v_{K+1} \in \mathbb{R}}} \Big\{ \sum_{k=1}^{K+1} [C_{\tau_{k-1}:\tau_k}(p_{k-1}, p_k, v_k) + \beta] \Big\},$$

$$s.t. \quad v_{k-1} = 2\frac{p_{k+1} - p_k}{\tau_{k+1} - \tau_k} - v_k, \forall k = 1, \dots, K+1,$$

where the minimum is taken over the unknown change-point vector $\boldsymbol{\tau} = \{\tau_k\}_{k=0}^{K+1}$ of unknown length $K+2$, the unknown position vector $\{p_k\}_{k=0}^{K+1}$, and the unknown final speed v_{K+1}. Specifying these quantities uniquely determines a quadratic spline (see Appendix F).

We emphasize that, for any generic $\boldsymbol{\tau} \in \mathcal{T}_t$, we denote its length by $K+2$, removing its dependency on the choice of vector in $\mathcal{T}_t$. Recall that we use the shorthand notations $p_i = p_{\tau_i}$ and $v_i = v_{\tau_i}$. We now introduce a functional cost $Q^{(t)}(p, v)$, temporarily leaving the optimization with respect to the parameters $p = p_{K+1}$ and $v = v_{K+1}$ aside. We obviously get $Q^{(t)} = \min_{p,v}\{Q^{(t)}(p, v)\}$.

The result is obtained by conditioning the sum over the position in time and space of the last change point. We write:

$$Q^{(t)}(p, v) = \min_{\substack{\boldsymbol{\tau}\in\mathcal{T}_t \\ \boldsymbol{p}\in\mathbb{R}^{K+1}}} \left\{ \sum_{k=1}^{K} [C_{\tau_{k-1}:\tau_k}(p_{k-1}, p_k, v_k) + \beta] + C_{\tau_K:\tau_{K+1}}(p_K, p, v) + \beta \right\},$$

$$s.t. \quad p = p_{K+1}, \quad v = v_{K+1}, \quad v_{k-1} = 2\frac{p_{k+1} - p_k}{\tau_{k+1} - \tau_k} - v_k, \forall k = 1, \dots, K+1,$$

and the minimization can be organized as follows:

$$\min_{\substack{0\le s<t \\ \tilde{p}\in\mathbb{R}}} \left\{ \min_{\substack{\boldsymbol{\tau}\in\mathcal{T}_s, \tau_K = s \\ \boldsymbol{p}\in\mathbb{R}^{K+1} \\ p_{K+1}=\tilde{p}}} \left\{ \sum_{k=1}^{K} [C_{\tau_{k-1}:\tau_k}(p_{k-1}, p_k, v_k) + \beta] \right\} + C_{s:t}(\tilde{p}, p, v) + \beta \right\}.$$

The inner minimum taken with its backward speed relations is exactly given by the value

$$Q^{(s)}(\tilde{p}, v_K) = Q^{(s)}\left(\tilde{p}, 2\frac{p - \tilde{p}}{t - s} - v\right),$$

and we obtain

$$Q^{(t)}(p, v) = \min_{\substack{0\le s<t \\ \tilde{p}\in\mathbb{R}}} \left\{ Q^{(s)}\left(\tilde{p}, 2\frac{p - \tilde{p}}{t - s} - v\right) + C_{s:t}(\tilde{p}, p, v) + \beta \right\},$$

which proves the result.

Appendix C. Proof of Proposition 3 (non-optimality)

We consider the data $y_0 = 0$, $y_1 = 1$, and $y_2 = 2 = y_3 = y_4 = \dots$. Let the sets of states be $\mathcal{P}_t = \{y_t\}$ for all t and the initial speeds set $\mathcal{V}_0 = \{1, 2\}$.

With data y_0, y_1, y_2, the best solution is the straight line $f(t) = t$, resulting in a cost of β. This is the minimal possible value for any segmentation. At this time, the second best solution is $\tilde{f}(t) = \frac{t(4-t)}{2}$ with as cost of $\beta + 1/4$. with initial speed $\tilde{f}'(0) = 2$. Notice that $\tilde{f}(2) = 2$, and its outer speed is zero. At all further times, completing the solution $\tilde{f}$ by the constant 2 leads to a cost of $2\beta + 1/4$. Completing the solution f with $t \ge 4$ leads to a higher cost due to the outer speed of 1 at time 2 when $\beta > \frac{1}{4}$. Thus, the best solution for $t \ge 4$ should be $2\beta + \frac{1}{4}$.

It is possible to show that all other solutions with no change have a cost greater than $2\beta + \frac{1}{4}$ for $t \ge 7$ when $\beta = \frac{1}{4} + \epsilon$. This proves our result.

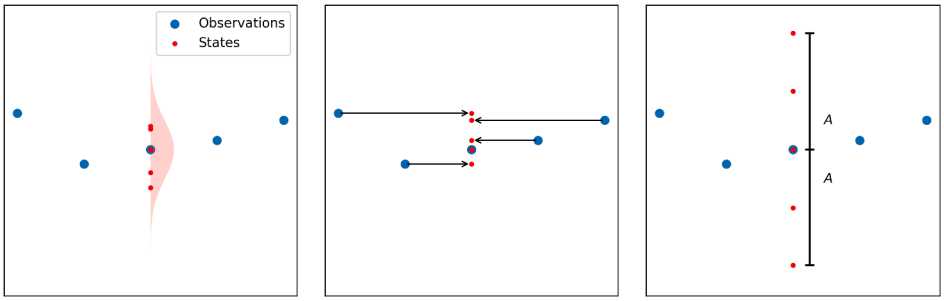
Appendix D. Discretization of positions

The discretized positions are meant to approximate the true values of the signal. Ideally, in a noiseless situation, having $P_t = \{f_t\}$ is enough. When the signal is noisy, we add more values to P_t trying to compensate for the uncertainty of the noise. Three strategies were tested, all including the observation as one of the states. Call N the total number of states, then the strategies are:

- *Random*: States are sampled from a Gaussian distribution centered at the current observation and with variance equal to the estimated one of the signal;
- *Neighbors*: $(N-1)/2$ points before and after the current observation are used as states. Thus, the set is:
  $P_t = \{y_{t-(N-1)/2}, y_{t-(N-1)/2+1}, \cdots y_t, y_{t+1}, \cdots, y_{t+(N-1)/2}\}$;
- *Regular grid*: An interval centered at the observation is sampled regularly N times. In the experiment, the width of the interval was set to 10% the amplitude of the signal. Call $2A = 0.1|y_{\max} - y_{\min}|$ the length of the interval, then $P_t = \{y_t - A, \dots, y_t, \dots., y_t + A\}$.

Fig. D.9, below, shows a simple example in one dimension with 5 states.

Fig. D.10 shows average MSE over the synthetic dataset with the constrained SplineOP setting $K = K^*$ and varying the number of states and state generation strategy. Results show that the order of magnitude of the reconstruction error remains the same across different generation strategies. At the same time, there seems to be an ordering, where the random strategy outperforms the neighbor strategy which itself outperforms the regular grid.



(a) Random generation. States are sampled from a Gaussian centered at the current observation.

(b) Neighbor generation. States are sampled as immediately preceding and succeeding observed points.

(c) Regular sampling. States are sampled from a regular grid centered at the current observation.

Fig. D.9. Illustration of different strategies for the generation of discrete states.

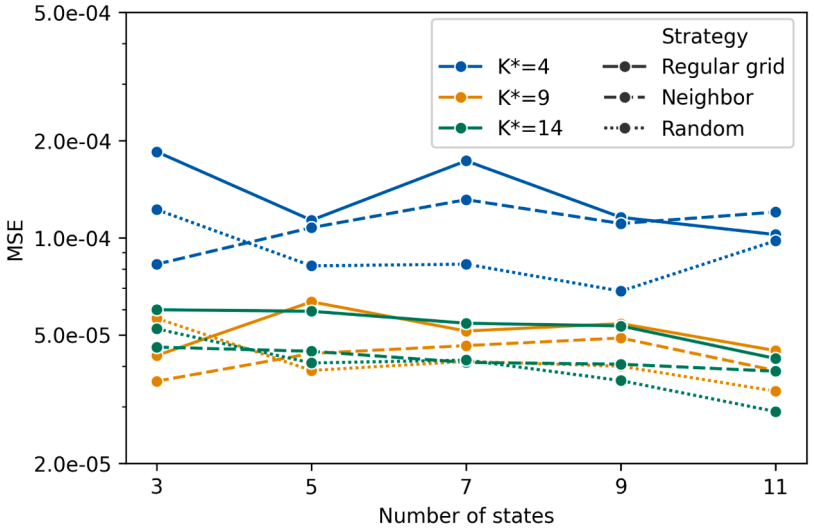


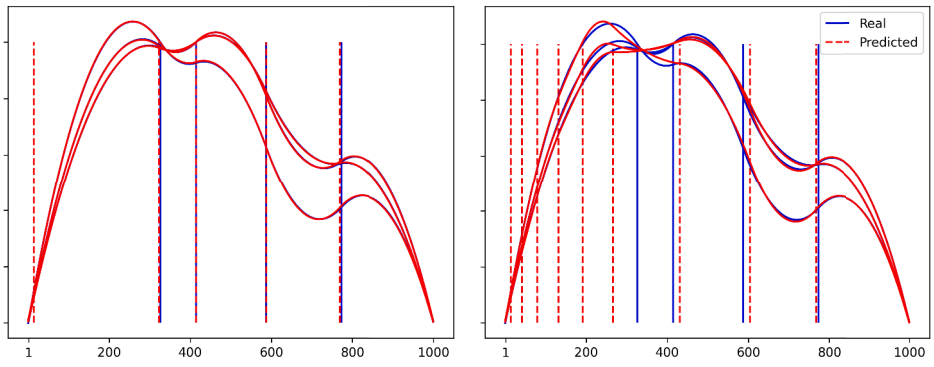
Fig. D.10. MSE comparison of different state generators for the constrained model (with $K = K^*$). Results show that the Random generator is superior to the others in most scenarios.

Appendix E. Robustness of the algorithm with respect to speed misspecification

Proposition 3 shows that the algorithm is not globally optimal due to its greedy speed selection strategy. Consequently, intermediate solutions may produce final speed estimates that are poorly suited to subsequent observations. To quantify the impact of such speed misspecification on performance, we conducted a series of new simulations with different levels of perturbation of the oracle speed. The results show that SplineOP has a compensation mechanism that, by introducing false positive changepoints, manages to correct the speed and perform well on the rest of the signal, as shown in Fig. E.11:

The experiments were run on the synthetic dataset with $K^* = 4$. Given the oracle initial speed $\upsilon = (\upsilon_x, \upsilon_y, \upsilon_z)$, we ran the algorithm with a single initial speed set to either the oracle speed or a perturbed version of it. To control the level of inaccuracy of the speed, we define the perturbation as 3D rotations. We treat υ as the axis of a cone. The vector υ is embedded within the cone and rotated around this axis. The angle defines the cone's half-opening angle (the tilt), while the second angle, specifies the rotation around the cone axis (the spin). In our experiments, we consider tilt angles of 0º, 15º, 30º, 45º, 60º, 75º, and 90º, and spin angles of 0º and 180º.

Aggregate performance metrics, in Fig. E.12, illustrate the effect of these perturbations and show that the pattern shown in Fig. E.11 is general: Fig. E.12a show an increase in the average number of predicted changepoints, which has a significant negative effect on the F-Score, as shown in Fig. E.12b. The MSE (Fig. E.12c) also deteriorates, but remains of the same order of magnitude (i.e. $O(10^{-4})$).



(a) Initial speed is the Oracle with a tilt of 15º. (b) Initial speed is the Oracle with a tilt of 90º.

Fig. E.11. Example of correcting behaviour of SplineOP for (a) small and (b) very big perturbation of the initial speed. Vertical lines indicate the predicted (dashed, red) and real (continuous, blue) changepoint positions. The predicted signal (red) has many more changepoints in the early stages than the real one (blue). SplineOP incorporates many false positive changepoints to stabilize the speed into the correct range of values.

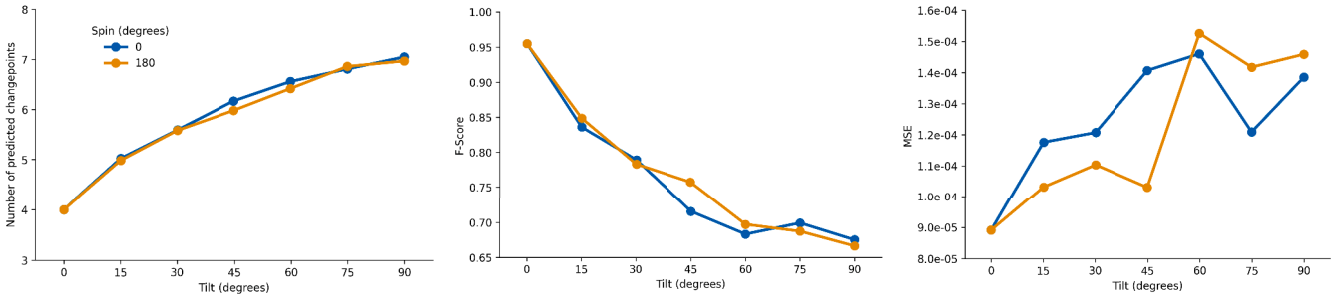


Fig. E.12. The number of predicted changepoints (E12.a) augments as the perturbation of the initial speed increases. This is a compensating mechanism of SplineOP that has a negative effect on the F-Score (E12.b) and MSE (E12.c). The latter one being less significant, as the predicted polynomial remains close to the observed points. .

Appendix F. Equations and unknowns in spline inference

A spline with K knots is described by the following $K+1$ quadratics:

$$f_k(t) = \frac{a_k}{2}(t-\tau_k)^2 + v_k(t-\tau_k) + p_k, \quad k = 1, \dots, K+1, \ t \in [\tau_{k-1}, \tau_k),$$

with $\tau_0 = 0$ and $\tau_{K+1} = T$. The vector of change points (knots) is an increasing vector of integers: $0 = \tau_0 < \tau_1 < \dots < \tau_K < \tau_{K+1} = T$. Continuity of the spline value and its derivative leads to the following system of equations:

$$\begin{cases} p_k = \dfrac{a_{k+1}}{2}(\tau_k - \tau_{k+1})^2 + v_{k+1}(\tau_k - \tau_{k+1}) + p_{k+1}, \\ v_k = a_{k+1}(\tau_k - \tau_{k+1}) + v_{k+1}, \end{cases} \tag{F.1}$$

for $k \in \{0, \dots, K\}$, considering additional starting elements p_0 and v_0. This can be simplified the following $2(K+1)$ equations:

$$\begin{cases} a_{k+1} = \dfrac{v_{k+1} - v_k}{\tau_{k+1} - \tau_k}, \\ \dfrac{p_{k+1} - p_k}{\tau_{k+1} - \tau_k} = \dfrac{v_{k+1} + v_k}{2}. \end{cases} \tag{F.2}$$

These equations are translated into implicit transparent notations: $A = \Delta V$ and $\overline{V} = \Delta P$. We introduce the notation $\Delta\tau_k = \tau_k - \tau_{k-1}$, used later on. Notice that notations a_k, v_k, and p_k are the acceleration (second derivative) the speed (first derivative) and the position (spline value) at time step τ_k, respectively.

Numbers of unknowns and equations.

- We have $3(K+1)$ unknowns with the quadratics (a_k, v_k, p_k) and the initial position and speed: p_0 and v_0. We obtain $3(K+1)+2$ unknowns;
- We have $2(K+1)$ equations with (F.2). Consider that the positions $p_0, \dots, p_{K+1}$ are explicitly or implicitly (by some equations) known. Same with the initial speed v_0. We obtain $2(K+1)+(K+2)+1 = 3(K+1)+2$ equations.

Observe that specifying a single speed value is enough; this may be v_0 or another value at a change point location.

For spline generation, we can also determine the spline by specifying all the acceleration parameters, $a_1, \dots, a_{K+1}$, and two positions, for example, setting $p_0 = p_{K+1} = 0$. This also gives the necessary $K+3$ additional relations.

Appendix G. Spline inference: from accelerations to positions

Consider the case with given accelerations and specified initial and final positions. This forms the standard simulation framework for generating quadratic spline data. Our algorithm identifies discontinuities (jumps) in acceleration, which are the key quantities to be controlled. From the accelerations and the positions of the knots, we can derive the speeds, thereby reformulating the spline constraints as a linear system, $Av = b$:

$$A = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \\ \delta_0 & \delta_1 & \cdots & \delta_K & \delta_{K+1} \end{pmatrix}_{(K+2)\times(K+2)},$$

with $\delta_0 = \frac{1}{2}\Delta\tau_1$, $\delta_{K+1} = \frac{1}{2}\Delta\tau_{K+1}$ and $\delta_i = \frac{1}{2}(\Delta\tau_i + \Delta\tau_{i+1})$ for i in $\{1, \dots, K\}$, and

$$v = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_K \\ v_{K+1} \end{pmatrix}, \; b = \begin{pmatrix} a_1\Delta\tau_1 \\ a_2\Delta\tau_2 \\ \vdots \\ a_{K+1}\Delta\tau_{K+1} \\ p_{K+1} - p_0 \end{pmatrix}.$$

Here the last equation is derived from the sum of all the elements of the second equation in (F.2). We used here the knowledge of p_0 and p_{K+1}. Eventually, the spline positions are given for $k = 0, \dots, K$ by relations:

$$p_{k+1} = \frac{1}{2}\sum_{i=0}^{k}(v_i + v_{i+1})(\tau_{i+1} - \tau_i) + p_0\,.$$

Thus, all parameters (a_k, v_k, p_k) are known.

Proposition 4. *(Spline definition with known accelerations) We consider the accelerations a_k as given in all segments (for $k = 1, \dots, K+1$) with also known initial and final positions p_0 and p_{K+1}. We introduce notations:*

$$\Delta\tau_k = \tau_k - \tau_{k-1}, \quad \bar{\tau}_k = \frac{1}{2}(\tau_k + \tau_{k-1}) \quad \textit{and} \quad \bar{v}_k = \frac{1}{2}(v_k + v_{k-1})\,.$$

The speeds and positions at the knots can be explicitly found. Their values are:

$$v_k = v_{K+1} - \sum_{j=k+1}^{K+1} a_j\Delta\tau_j, \quad \textit{for all } k \in \{0, \dots, K\},$$

where

$$v_{K+1} = \frac{p_{K+1} - p_0}{T} + \frac{1}{T}\sum_{j=1}^{K+1} a_j(\bar{\tau}_j\Delta\tau_j),$$

and

$$p_k = p_0 + \sum_{i=1}^{k}\bar{v}_i\Delta\tau_i = p_{K+1} - \sum_{i=k+1}^{K+1}\bar{v}_i\Delta\tau_i\,, \quad \textit{for all } k \in \{1, \dots, K\}\,.$$

Proof. Given the sparsity of the matrix A some algebraic work leads to an explicit inverse:

$$A^{-1} = \begin{pmatrix} d_0 & d_1 & d_2 & d_3 & \cdots & 1/T \\ q_0 & d_1 & d_2 & d_3 & \cdots & \vdots \\ \vdots & q_1 & d_2 & d_3 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ q_0 & q_1 & \cdots & \cdots & d_K & 1/T \\ q_0 & q_1 & \cdots & \cdots & q_K & 1/T \end{pmatrix}.$$

The terms under the diagonal q_k are:

$$q_k = \frac{\sum_{i=1}^{k}\Delta\tau_i + \Delta\tau_{k+1}/2}{\sum_{i=1}^{K+1}\Delta\tau_i} = \frac{(\tau_k - \tau_0) + \Delta\tau_{k+1}/2}{\tau_{K+1} - \tau_0} = \frac{\frac{\tau_{k+1}+\tau_k}{2} - \tau_0}{\tau_{K+1} - \tau_0} = \frac{\bar{\tau}_{k+1}}{T},$$

and the terms on the diagonal and above are given by $d_k = -(1 - q_k)$. From these results, we can write each component of the speed vector $\boldsymbol{v} = (v_1, \dots, v_{K+1})$ as:

$$v_k = \frac{p_{K+1} - p_0}{T} + \sum_{i=0}^{k-1} q_i a_{i+1}\Delta\tau_{i+1} + \sum_{j=k}^{K} d_j a_{j+1}\Delta\tau_{j+1},$$

or, by first computing v_{K+1}, we realize that:

$$v_k = v_{K+1} - \sum_{j=k}^{K} a_{j+1}\Delta\tau_{j+1}\,.$$

Once the velocities are known, recalling that p_0 and p_{K+1} are given, we write the system of equations that arises from the second equation in (F.2): $Bp = \bar{v}$

$$B = \begin{pmatrix} 1 & 0 & \cdots & \cdots & & \cdots & 0 \\ -1 & 1 & 0 & 0 & \cdots & & 0 \\ 0 & -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & -1 & 1 & 0 & 0 \\ 0 & & & \cdots & -1 & 1 & 0 \\ 0 & & & \cdots & 0 & -1 & 1 \\ 0 & & & \cdots & & 0 & 1 \end{pmatrix}_{(K+3)\times(K+2)},$$

$$p = \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ \vdots \\ p_{K+1} \end{pmatrix}, \text{ and } \bar{v} = \begin{pmatrix} p_0 \\ \bar{v}_1\Delta\tau_1 \\ \vdots \\ \bar{v}_{K+1}\Delta\tau_{K+1} \\ p_{K+1} \end{pmatrix},$$

which can be solved by forward or backward substitution, yielding:

$$p_k = p_0 + \sum_{i=1}^{k}\bar{v}_i\Delta\tau_i\,, \quad \text{for all } k \in \{1, \dots, K\},$$

or

$$p_k = p_{K+1} - \sum_{i=k+1}^{K+1}\bar{v}_i\Delta\tau_i\,, \quad \text{for all } k \in \{1, \dots, K\}\,.$$

□

Appendix H. Spline inference: from positions to accelerations

We consider an oracle situation for which the position of the knots is known. We derive all the parameters of the spline as functions of the initial speed v_0.

Proposition 5. *Given the positions p_k for $k = 0, \dots, K+1$ and the initial speed v_0, we compute the speed and acceleration vectors, emphasizing their dependence on the initial speed. We obtain:*

$$v_k(v_0) = 2\sum_{i=1}^{k}(-1)^{k-i}\frac{p_i - p_{i-1}}{\tau_i - \tau_{i-1}} + (-1)^k v_0, \quad k \in \{1, \dots, K+1\}$$

and

$$a_k(v_0) = \frac{2}{\tau_k - \tau_{k-1}}\sum_{i=1}^{k}(-1)^{k-i}\frac{p_i - p_{i-1}}{\tau_i - \tau_{i-1}} + \frac{2(-1)^k}{\tau_k - \tau_{k-1}}v_0, \; k \in \{1, \dots, K+1\}\,.$$

Proof. Using the second equation in (F.2) we get the system of equations $Av = b$, where:

$$A = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & 1 \end{pmatrix} \quad \text{and} \quad v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{K+1} \end{pmatrix}, \; b = \begin{pmatrix} 2\frac{p_1 - p_0}{\tau_1 - \tau_0} - v_0 \\ 2\frac{p_2 - p_1}{\tau_2 - \tau_1} \\ \vdots \\ 2\frac{p_{K+1} - p_K}{\tau_{K+1} - \tau_K} \end{pmatrix}.$$

And from all the speeds v, we easily get the accelerations a. □

All coefficients $v_k(v_0)$ and $a_k(v_0)$ linearly depend on v_0 and on the positions p_k.

References

[1] Y.V. Parkale, S.L. Nalbalwar, Compressed sensing for ECG signal compression using DWT based sensing matrices, Smart Sci. 11 (4) (2023) 759–773.

[2] M.H. Trabuco, M.V.C. Costa, B. Macchiavello, F.A.d.O. Nascimento, S-EMG Signal compression in one-dimensional and two-dimensional approaches, IEEE J. Biomed. Health Inform. 22 (4) (2017) 1104–1113.

[3] F. Eichinger, P. Efros, S. Karnouskos, K. Böhm, A time-series compression technique and its application to the smart grid, VLDB J. 24 (2) (2015) 193–218.

[4] D. Blalock, S. Madden, J. Guttag, Sprintz: time series compression for the internet of things, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2 (3) (2018) 1–23.
[5] M.A. de Oliveira, A.M. da Rocha, F.E. Puntel, G.G.H. Cavalheiro, Time series compression for IoT: a systematic literature review, Wirel. Commun. Mob. Comput. 2023 (1) (2023) 5025255.
[6] R. Sabbagh, Z. Cai, A. Stothert, D. Djurdjanovic, Physically inspired data compression and management for industrial data analytics, Front. Comput. Sci. 2 (2020) 41.
[7] H. Vedam, V. Venkatasubramanian, M. Bhalodia, A B-spline based method for data compression, process monitoring and diagnosis, Comput. Chem. Eng. 22 (1998) S827–S830.
[8] M. Karczewicz, M. Gabbouj, ECG Data compression by spline approximation, Signal Process. 59 (1) (1997) 43–59.
[9] G. Chiarot, C. Silvestri, Time series compression survey, ACM Comput. Surv. 55 (10) (2023) 1–32.
[10] T. Lyche, C. Manni, H. Speleers, Foundations of Spline Theory: B-Splines, Spline Approximation, and Hierarchical Refinement, Springer International Publishing, 2018. https://doi.org/10.1007/978-3-319-94911-6_1
[11] C. De Boor, On uniform approximation by splines, J. Approx. Theory 1 (1) (1968) 219–235.
[12] G. Wahba, Spline Models for Observational Data, SIAM, 1990.
[13] P.H.C. Eilers, B.D. Marx, Splines, knots, and penalties, WIREs Comput. Stat., 2 (6) (2010) 637–653. https://doi.org/10.1002/wics.125
[14] G. Beliakov, Least squares splines with free knots: global optimization approach Appl. Math. Computat. 149 (3) (2004) 783–798. https://doi.org/10.1016/S0096-3003(03)00179-6
[15] D.L.B. Jupp, Approximation to Data by Splines with Free Knots, SIAM Journal on Numerical Analysis 15 (2) (1978) 328–343. http://www.jstor.org/stable/2156756
[16] H.G. Burchard, Splines (with optimal knots) are better, Applicable Analysis 3 (4) (1974) 309–319. https://doi.org/10.1080/00036817408839073
[17] V.T. Dung, T. Tjahjowidodo, A direct method to solve optimal knots of B-spline curves: an application for non-uniform B-spline curves fitting, PloS one 12 (3) (2017) e0173857.
[18] F. Yao, T.C.M. Lee, On knot placement for penalized spline regression, J. Korean Stat. Soc. 37 (3) (2008) 259–267.
[19] C.H. Reinsch, Smoothing by spline functions. II Numer. Math. 16 (5) (1971) 451–454. https://doi.org/10.1007/BF02169154
[20] C.R. Rojas, B. Wahlberg, On change point detection using the fused lasso method, (2014) arXiv preprint arXiv:1401.5408.
[21] C. Wen, X. Wang, A. Zhang, l0 trend filtering, INFORMS J. Comput. 35 (6) (2023) 1491–1510.
[22] J. Shi, F. Wang, M. Qin, A. Chen, W. Liu, J. He, H. Wang, S. Chang, Q. Huang, New ECG compression method for portable ECG monitoring system merged with binary convolutional auto-encoder and residual error compensation, Biosensors 12 (7) (2022) 524.
[23] D. Hsu, Time series compression based on adaptive piecewise recurrent autoencoder, (2017) arXiv preprint arXiv:1707.07961.
[24] T.D. Hocking, G. Rigaill, P. Fearnhead, G. Bourque, Generalized functional pruning optimal partitioning (GFPOP) for constrained changepoint detection in genomic data, J. Stat. Softw. 101 (2022) 1–31.
[25] V. Runge, T.D. Hocking, G. Romano, F. Afghah, P. Fearnhead, G. Rigaill, gfpop: an R package for univariate graph-constrained change-point detection, J. Stat. Softw. 106 (2023) 1–39.
[26] P. Fearnhead, R. Maidstone, A. Letchford, Detecting changes in slope with an L0 penalty, J. Comput. Graph. Stat. 28 (2) (2019) 265–275.
[27] V. Runge, M. Pascucci, N.D. de Boishebert, Change-in-slope optimal partitioning algorithm in a finite-size parameter space, (2020) arXiv preprint arXiv:2012.11573.
[28] R. Killick, P. Fearnhead, I.A. Eckley, Optimal detection of changepoints with a linear computational cost, J. Am. Stat. Assoc. 107 (500) (2012) 1590–1598.
[29] R. Maidstone, T. Hocking, G. Rigaill, P. Fearnhead, On optimal multiple changepoint algorithms for large data, Stat. Comput. 27 (2) (2017) 519–533.
[30] P. Hall, J.W. Kay, D.M. Titterington, Asymptotically optimal difference-based estimation of variance in nonparametric regression, Biometrika 77 (3) (1990) 521–528.
[31] S.-J. Kim, K. Koh, S. Boyd, D. Gorinevsky, \ell_1 trend filtering, SIAM Rev. 51 (2) (2009) 339–360.
[32] N. Cecchi, V. Runge, C. Truong, L. Oudre, SPOP: time series compression with quadratic splines, in: Proceedings of the 33rd European Signal Processing Conference, 2025, pp. 2687–2692.
[33] S.W. Combettes, P. Boniol, A. Mazarguil, D. Wang, D. Vaquero-Ramos, M. Chauveau, L. Oudre, N. Vayatis, P.-P. Vidal, A. Roren, et al., Arm-CODA: a data set of upper-limb human movement during routine examination, Image Process. Line 14 (2024) 1–13.