SPOP: Time Series Compression With Quadratic Splines

Nicolás Enrique Cecchi[1], Vincent Runge[2], Charles Truong[1], Laurent Oudre[1]
[1]*Université Paris Saclay, Université Paris Cité, ENS Paris Saclay, CNRS, SSA, INSERM, Centre Borelli,* F-91190, Gif-sur-Yvette, France
[2]*Université Paris Saclay, Université d'Evry Val d'Essonne, CNRS, LaMME,* F-91037, Evry, France

***Abstract*—Time series compression is an important topic, enabling long signals to be represented in a small number of parameters while denoising. Change-point detection methods offer a relevant framework for tackling this task. In this paper, we propose a new algorithm to approximate time series as piecewise quadratic functions with continuity constraints. We propose a dynamic programming recursion to compute the optimal solution to this problem and an approximation algorithm with quadratic complexity that locally discretizes the spaces of admissible values at change points and initial first derivatives. We validate its performance on signals from a real data set and on synthetic signals.**

***Index Terms*—change-point detection, compression, time series, splines, dynamic programming.**

I. Introduction

Time series are recorded in numerous fields of study, such as medicine, sports, or manufacturing [1]. Time series data are often noisy and can be very large, especially when collected at high sampling frequencies. Compressing time series and approximating them with parametric functions provides a relevant solution for their study, allowing both the representation of the information they contain with few parameters (that can be used as input features for machine learning tasks) and the ability to perform denoising.

One solution to compute this compression is to rely on change point detection: given a time series $\{y_i\}_{i=1}^{T}$, we search a vector $\hat{\tau} = \{\hat{\tau}_0 = 1 < \hat{\tau}_1 < \cdots < \hat{\tau}_K = T\}$ that partitions the signal into K contiguous segments $[\hat{\tau}_k, \hat{\tau}_{k+1})_{i=0}^{K-1}$, that can each be modelled with a parametric function. In the context of splines, change points are also called knots. For K segments, there exist $\binom{T-2}{K-1}$ possible segmentations; therefore, a brute-force approach quickly becomes impracticable. Numerous efficient algorithms exist to estimate the number and locations of changes accurately. The most efficient strategies rely on dynamic programming [2] or binary segmentation [3]. Different parametric functions have been investigated in the literature: piecewise constant [4] and piecewise linear functions [5] are among the most popular. In many contexts, it makes sense to exploit the dependencies that exist between successive segments. To this end, some authors have introduced continuity constraints, notably for piecewise linear approximations [6], [7].

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

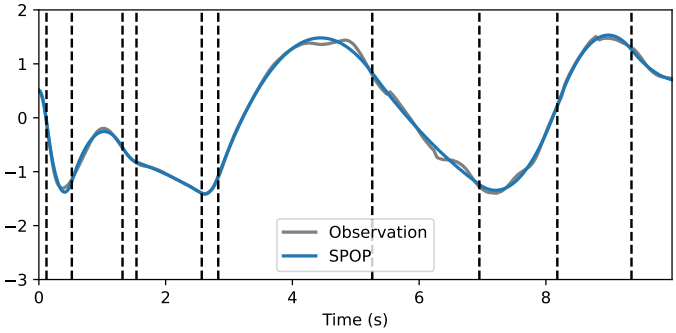


Fig. 1: Two examples from a motion capture dataset [1]. Our approximation is a quadratic spline; vertical dashed lines indicate knot positions found by our method SPOP.

In this work, we consider the problem of piecewise quadratic function approximation (continuous first derivative and piecewise constant second derivative) with continuity constraints. This problem is of practical use in the analysis of smooth trajectory data, and can provide better compression/approximation than the widely studied piecewise linear case. Our algorithm is available in an online repository[1].

a) Related work: A well-known approach to approximate smooth time series is ℓ_1 trend filtering [8] which solves

$$\min_{(u_t)_t} \sum_{t=1}^{T} ||y_t - u_t||_2^2 + \lambda ||D^{(3)} u||_1 \qquad (1)$$

with $D^{(3)}$ the 3^{rd}-discrete difference operator, $\|\cdot\|_1$ the ℓ_1 norm and $\lambda > 0$ a user-defined parameter. By design, ℓ_1 trend filtering approximation is close to the observations and has few jumps in the 2nd derivative. However, because the ℓ_1 norm penalizes the absolute values of changes, solutions suffer from over-segmentation: instead of a few significant changes, several small ones are detected. This effect is also known as the staircase effect in the Lasso regression [9]

For sparser results, the authors of [10] replace in (1) the ℓ_1 norm by the ℓ_0 norm:

$$\min_{(u_t)_t} \sum_{t=1}^{T} ||y_t - u_t||_2^2 \quad \text{s.t.} \quad ||D^{(3)} u||_0 \leq K. \qquad (2)$$

[1]https://github.com/nicolascecchi/splineop

The ℓ_0 norm counts the number of non-zero elements. This problem is untractable, so the authors describe an alternating minimization scheme.

In [7], the authors fit a continuous piecewise linear function. To simplify the problem, they constrain the approximating functions to take values in a finite – typically small – set at the change points. The proposed algorithm relies on this simplification and dynamic programming. Our approach extends this method to the piecewise quadratic setting.

b) Our contribution: We present a method for compressing signals using a quadratic spline model. The knot positions are estimated with a change point detection algorithm. The complexity of the proposed approach is quadratic in the number of observations. Compared to baselines, our adaptive knot estimation procedure empirically produces better approximations with fewer knots on synthetic and real-world data.

II. Problem statement

a) Signal model: Consider observations $\{y_t\}_{t=1}^T \in \mathbb{R}^T$ sampled at successive times $t = 1, 2, \dots, T$ modeled as:

$$y_t = f^\star(t) + \epsilon_t\,, \quad t = 1, \dots, T\,,$$

where the $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ are iid with variance σ^2 and $f^\star : [0, +\infty) \to \mathbb{R}$ is an unknown smooth function to estimate. The function $f^\star$ follows a quadratic spline model, meaning that:

- there exist $K^\star - 1$ indices $\tau_k^\star$ and K^* triples $(a_k^\star, b_k^\star, c_k^\star)$ such that for $t \in [\tau_k^\star, \tau_{k+1}^\star)$:

$$f^\star(t) = a_k^\star t^2 + b_k^\star t + c_k^\star\,, \quad k = 0, \dots, K^\star - 1\,,$$

  where $\tau_0^\star = 1$ and $\tau_{K^\star}^\star = T$, by convention;
- the function $f^\star$ is continuously differentiable, i.e. $f^\star$ is differentiable and $f^{\star\prime}$ is continuous.

Consequently, the acceleration $f^{\star\prime\prime}$ is piecewise constant. The change points are the indices $\tau_k^\star$ where the jumps of $f^{\star\prime\prime}$ occur, meaning $f^{\star\prime\prime}(\tau_k^{\star-}) \neq f^{\star\prime\prime}(\tau_k^{\star+})$.

b) Optimization problem: Define $\mathcal{S}_K$ as the set of all quadratic splines with $K - 1$ knots, or equivalently, K segments where the function follows a quadratic form. A common approach to change-point detection involves minimizing the reconstruction error:

$$Q_T^K := \min_{f \in \mathcal{S}_K} \Big\{ \sum_{t=1}^{T} (y_t - f(t))^2 \Big\}\,. \tag{3}$$

Solving this optimization problem is challenging, even for short signal lengths [6], [7]. The main difficulty arises from the continuity constraints at the knots, which hinder the application of standard dynamic programming techniques.

In this work, we assume that the number of segments K to be detected is fixed in advance and remains small relative to the signal length. Even if the optimal K is unknown, one can resort to heuristics such as the slope heuristic, the elbow method, or cross-validation to select an appropriate value. An alternative approach that minimizes a penalized reconstruction error [11], [12] is left for future work.

We focus on the regularly sampled setting for simplicity. We can straightforwardly extend our approach to the case of irregularly sampled time series.

III. Our approach

Our approach consists of two key methodological steps:

- We establish a recursive formulation for the objective function (3) using dynamic programming, though it remains intractable;
- To approximate this recursion, we (i) constrain the splines to a finite set of possible values at the knots and (ii) restrict initial speeds to a finite discrete set.

a) Dynamic programming: First, notice that the total reconstruction error of a spline is the sum of K terms; each term is equal to the reconstruction error on one segment $[t_{\tau_k}, t_{\tau_{k+1}})$. Dynamic programming iteratively computes the cost of fitting $1, \dots, K$ segments to the data. An efficient algorithm needs a recursive relationship from k to $k + 1$ segments.

We formally introduce the cost of fitting a polynomial of order two on a segment to derive this relationship. For two indices $a < b$, an initial position p_a, a final position p_b and a final speed v_b, there exists a unique quadratic function $f(x) = \alpha x^2 + \beta x + \gamma$ such that $f(a) = p_a$, $f(b) = p_b$ and $f'(b) = v_b$. Then, the reconstruction error between indices a (included) and b (excluded) is $C_{a:b}(p_a, p_b, v_b)$ defined by:

$$C_{a:b}(p_a, p_b, v_b) := \sum_{t=a}^{b-1} \big(y_t - f(t)\big)^2\,. \tag{4}$$

A few algebraic manipulations using Faulhaber's formula [13] yield the following lemma.

Lemma 1. *Assume that $\sum_{s \le t} y_s$, $\sum_{s \le t} y_s^2$, $\sum_{s \le t} s y_s$, and $\sum_{s \le t} s^2 y_s$ are pre-computed for any $t \le T$, which can be done in $\mathcal{O}(T)$ operations. Then, for fixed parameters (a, b, p_a, p_b, v_b), the value of the segment cost $C_{a:b}(p_a, p_b, v_b)$ (4) can be computed in $\mathcal{O}(1)$ operations.*

An essential property of quadratic splines is that they are entirely characterized by their initial speed, knots, and positions at the knots. More precisely, let f be a quadratic spline with initial speed $f'(1) = v_0$, knot indices τ_k and positions $p_k = f(\tau_k)$ at the knots. Since f is polynomial of order 2 on $[\tau_0, \tau_1)$, the only possible value for $f'(\tau_1)$ is $v_1 = 2\frac{p_1 - p_0}{\tau_1 - \tau_0} - v_0$. By induction, $f'(\tau_k)$ is equal to $v_k = 2\frac{p_k - p_{k-1}}{\tau_k - \tau_{k-1}} - v_{k-1}$. Therefore, we can find the unique quadratic polynomial on each segment $[\tau_k, \tau_{k+1})$ using the initial position p_k, the final position p_{k+1} and initial speed v_k. The spline f is then wholly characterized.

Consequently, we rewrite the minimization (3) over $\mathcal{S}_K$ –the space of quadratic splines with K segments– as a minimization over the set of possible speeds and positions at the knots, with

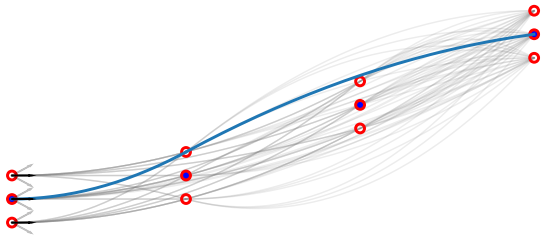


Fig. 2: Schematic view of different solutions (in light gray) evaluated by SPOP. The best one is in blue. Candidate splines all go through a red circle *if there is a knot* at this position. For instance, the blue spline does not go through a red circle on the third timestamp because the third timestamp is not a knot; only the second timestamp is a knot (as well as the first and last, by convention). Here, at each timestamp, SPOP considers three ($n = 3$) possible values; the actual observation is the middle point. Also, SPOP considers here three possible initial speeds (gray arrows). Note that not all candidate splines are shown for clarity.

the constraints that the speeds v_k satisfy the previous recursive relationship. Formally,

$$\begin{aligned} Q_T^K &= \min \left\{ \sum_{k=1}^{K} C_{\tau_{k-1}:\tau_k}\left(p_{k-1}, p_k, v_k\right) \right\} \\ \text{over } & v_0, \{\tau_k\}_{k=1}^{K-1}, \{p_k\}_{k=0}^{K}, \{v_k\}_{k=0}^{K} \\ \text{s.t. } & v_k = 2(p_k - p_{k-1})/(\tau_k - \tau_{k-1}) - v_{k-1}. \end{aligned} \tag{5}$$

We present a dynamic programming approach to find the minimizing value of (5). Let $Q_t^k(p, v)$ be the optimal reconstruction error when approximating the t first observations $\{y_s\}_{s=1}^t$ with a quadratic spline with k segments, and final position p and final speed v. We get the following result.

Proposition 1. *The following recurrence relation holds true for $k = 1, \ldots, K$ and $t = k+1, \ldots, T$:*

$$Q_t^k(p, v) = \min_{\substack{\tilde{p}\in\mathbb{R} \\ k\le s<t}} \left\{ Q_s^{k-1}\left(\tilde{p}, 2\frac{p-\tilde{p}}{t-s} - v\right) + C_{s:t}(\tilde{p}, p, v) \right\}, \tag{6}$$

with $Q_1^0(\cdot,\cdot) = 0$ and $Q_t^0(\cdot,\cdot) = +\infty$ for all $t > 1$. Also, $Q_T^K = \min_{p,v} Q_T^K(p, v)$ when Q_T^K is defined in (3) *or* (5).

Proof. For any $s < t$, we extend Q_s^{k-1} into Q_t^k by appending the last segment while ensuring continuity at index s for both position and derivative. The position continuity is achieved by optimizing over $\tilde{p}$, while the derivative continuity is enforced using the velocity uniquely determined by the relation $2\frac{p-\tilde{p}}{t-s} - v$, where p and v denote the position and speed at the end of the last segment. Indeed, the final polynomial on $[s, t)$ is $q(x) = (v(t-s) - (p-\tilde{p}))(\frac{x-s}{t-s})^2 - (v(t-s) - 2(p-\tilde{p}))\frac{x-s}{t-s} + \tilde{p}$. □

The function $Q_t^k(\cdot,\cdot)$ becomes increasingly complex as t grows. Currently, there is no efficient way to store it in memory. As a result, solving the recurrence of Prop. 1 remains intractable.

Algorithm 1 SPOP algorithm

```
# Compute one-segment solutions. Use set of initial speeds.
for t = 2 to T do                                   ▷ Loop over time
    for j = 1 to n do                               ▷ Loop over ending states
        C^1_{j,t} = { C_{1:t}(p_{1,i}, p_{t,j}, 2(p_{t,j} - p_{1,i})/(t-1) - v_{0,l}), ∀i, ∀l }
        Q^1(j,t) = min_{1≤i≤n, 1≤l≤m} C^1_{j,t}
        B^1(j,t) = ( argmin_{1≤i≤n} min_{1≤l≤m} C^1_{j,t}, 1 )
        V^1(j,t) = 2(p_{t,j} - p_{1,i})/(t-1) - v_{0,l*}  with l* argmin in l line 4
    end for
end for
# Compute k=2..K segments, propagating previous speeds.
for k = 2 to K do                                   ▷ Loop over number of segments.
    for t = k+1 to T do                             ▷ Loop over time.
        for j = 1 to n do                           ▷ Loop over ending states
            C^k_{j,t} = { Q^{k-1}(i,s)
                          + C_{s:t}(p_{s,i}, p_{t,j}, v(j,t)), ∀i, ∀s }
                        with v(j,t) = 2(p_{t,j} - p_{s,i})/(t-s) - V^{k-1}(i,s)
            Q^k(j,t) = min_{1≤i≤n, k≤s<t} C^k_{j,t}
            B^k(j,t) = (I,S) = argmin_{1≤i≤n, k≤s<t} C^k_{j,t}
            V^k(j,t) = 2(p_{t,j} - p_{s,I})/(t-S) - V^{k-1}(I,S)
        end for
    end for
end for
return (Q^1, ..., Q^K, B^1, ..., B^K)
```

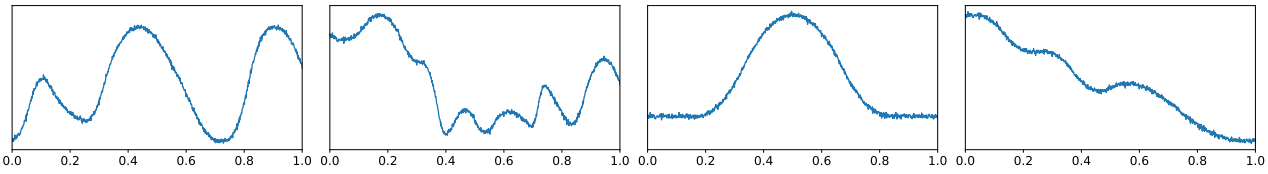
b) Approximate strategy: To circumvent the difficulty of dynamic programming, we restrict the initial speed v_0 and positions at knots p_t to a finite set of candidate values: $v_0 \in \mathcal{V}_0 = \{v_{0,1}, v_{0,2}, \ldots\}$ and $p_t \in \mathcal{P}_t = \{p_{t,1}, p_{t,2}, \ldots\}$. We describe later heuristics to choose appropriate sets $\mathcal{V}_0$ and $\mathcal{P}_t$. In other words, we only consider splines that can take a finite set of values at the knots and initial speeds. For more flexibility, the set of candidate values can change with the temporal position, meaning that $f(s)$ and $f(t)$ are approximated by different sets $\mathcal{P}_s$ and $\mathcal{P}_t$ if $t \neq s$. Fig. 2 shows a schematic view.

This discretization leads to the following approximation of the dynamic programming method (6) where $v_{p,t}^k$ is the approximate ending speed for a fit into k quadratics ending at time t and position p in P_t. The recursive formulation is then for $k \in \{1, \ldots, K\}$ and $t \in \{k+1, \ldots, T\}$ given by:

$$\begin{cases} Q_t^k(p, v_{p,t}^k) = \min\limits_{\substack{k\le s<t \\ \tilde{p}\in P_s}} \left\{ Q_s^{k-1}(\tilde{p}, v_{\tilde{p},s}^{k-1}) + C_{s:t}(\tilde{p}, p, v_{p,t}^k) \right\} \\ v_{p,t}^k = 2\frac{p-\tilde{p}}{t-s} - v_{\tilde{p},s}^{k-1}, \quad \text{with} \quad v_{p,1}^0 \in V_0\,. \end{cases} \tag{7}$$

For simplicity, we consider that all $\mathcal{P}_t$ have the same number n of elements, i.e. $n = |\mathcal{P}_t|$ for all t. The resulting iterative algorithm is called SPOP (SPline Optimal Partitioning); its pseudo-code is in Algorithm 1.

In the algorithm, each $\mathbf{C}_{j,t}^k$ is a set of costs over which we minimize. For $k = 1$, the minimization is over (i, j), the initial position and speed. For $k > 1$, over (i, s), the spatial and temporal position of the last change. Results for $Q_t^k(p, v_{p,t}^k)$



(a) Signal 1 (b) Signal 2 (c) Signal 3 (d) Signal 4

Fig. 3: Noisy synthetic signals

are saved into K matrices $\mathbf{Q}^k \in \mathbb{R}^{n\times T}$. We also need the "argument" matrices $\mathbf{B}^k$ in $(\{1,\dots,n\}\times\{1,\dots,T\})^{n\times T}$: $\mathbf{B}^k(j,t)$ stores the indices of the optimal state $p_{t,k}$ and time respectively for $\mathbf{Q}^k(j,t)$. We define the ending speed matrices $\mathbf{V}^k$ in $\mathbb{R}^{n\times T}$, used for propagation. We build the solution path $((p_{\hat{n}(0)},\hat{\tau}_0)\dots,(p_{\hat{n}(K)},\hat{\tau}_K))$ as follows:

$$\begin{cases}(\hat{n}(K),\hat{\tau}_K) = \big(\underset{j=1,\dots,n}{\operatorname{argmin}}\,\mathbf{Q}^K(j,T),T+1\big),\\ (\hat{n}(k),\hat{\tau}_k) = \mathbf{B}^{k+1}(\hat{n}(k+1),\hat{\tau}_{k+1}),\quad k=K-1,\dots,0.\end{cases}$$

Here, $\hat{n}(k)$ is the state index for knot k, i.e. $\hat{f}(\hat{\tau}_k) = p_{\hat{n}(k)}$ where $\hat{f}$ is the spline approximation returned by `SPOP`.

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

Proof. There are $3K$ matrices to fill ($\mathbf{Q}$, $\mathbf{B}$, $\mathbf{V}$), each of size $n\times T$. At time t, we compare at most $t\times n$ values in (7) in $\mathcal{O}(1)$ time (Lemma 1). For each matrix, we get a time upper bounded by $3\sum_{t=1}^{T}\sum_{j=1}^{n}(tn)\mathcal{O}(1)\leq\mathcal{O}(n^2T^2)$, which proves the result. □

c) In practice: The set of possible values at the knots is discretized by n uniformly spaced values, centered at the observation, covering a user-defined range (here, 5% of the signal's amplitude). We also consider several (here, 5) initial speeds, computed as the slopes of linear regressions over the first samples.

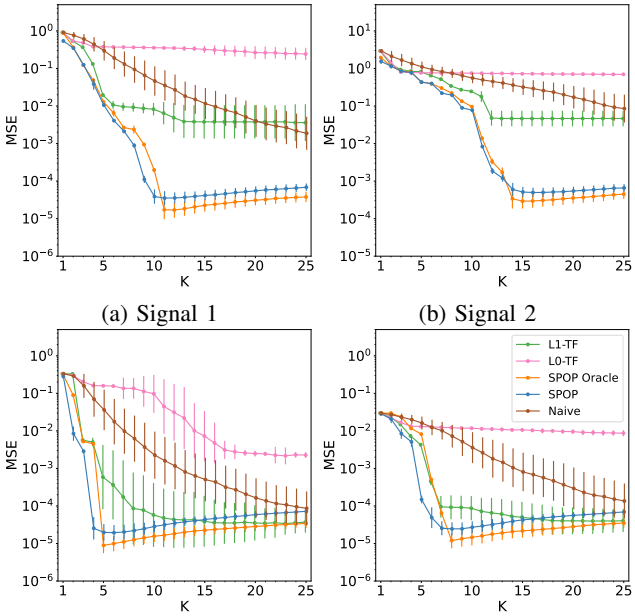
IV. Experiments

We compare `SPOP` to several baselines on synthetic and real data.

For ℓ_1 trend filtering (`L1-TF`), we use the implementation of [14]. We use a dichotomic search to find the correct penalty value for a given number of knots. For ℓ_0 trend filtering (`L0-TF`), we use the implementation of [10]. We also compare to a "naive" approach (`Naive`): we compute the discrete 3rd derivatives of the time series and consider values above a threshold as knots. Finally, for the synthetic data, our method is run with an oracle discretization (`SPOP Oracle`): the candidate state values are the true values at the knots, and the initial speed is the actual initial speed. It will serve as a bound on performance.

A. Synthetic data

a) Data: Four quadratic splines (Signals 1 to 4) are generated with $K^\star=11$, $K^\star=15$, $K^\star=5$ and $K^\star=8$ segments, respectively (see Fig. 3). For a number of samples T, each signal is sampled and added a Gaussian noise (SNR = 15 dB), this is repeated 100 times. Average MSE between the approximation and the true noiseless signal are reported.



(a) Signal 1 (b) Signal 2 (c) Signal 3 (d) Signal 4

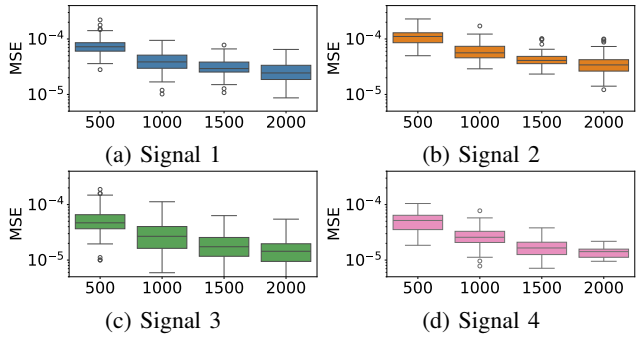
Fig. 4: MSE vs the number of detected changes K. Vertical bars indicate the variances over 100 realizations.

b) Influence of the compression rate on the MSE: Given signals of size $T=1000$, we apply the algorithms for a varying number of segments K. Fig. 4 shows the performances. We make several observations.

- For all signals, `SPOP` attains the minimum MSE at $K=K^*$. `SPOP` performs better than `SPOP Oracle` for $K<K^*$, and from $K\geq K^*$, the opposite is true.
- `L1-TF` performs reasonably well for Signals 3 and 4, which are simpler but considerably worse than `SPOP` on Signals 1 and 2, which are more complex.
- `Naive` and `L0-TF` do not perform well here.

Overall, `SPOP` and `SPOP Oracle` are the best performing methods, especially for small K, i.e. in high compression settings.

c) Influence of the sampling rate on the MSE: In this experiment, the number of segments to detect is known ($K=K^*$), and we study the MSE for varying signal lengths. Results are in Fig. 5. For all signals, `SPOP` better approximates when



(a) Signal 1 (b) Signal 2

(c) Signal 3 (d) Signal 4

Fig. 5: MSE of SPOP for varying signal lengths

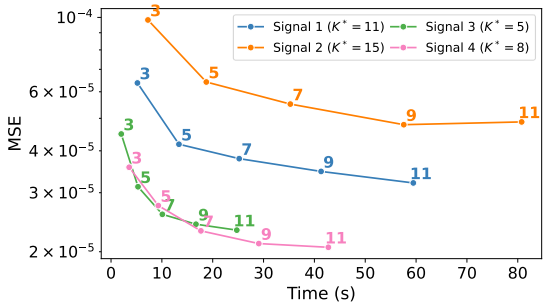


Fig. 6: MSE vs execution time for varying sizes n of the discrete state set. Above each point, n is shown.

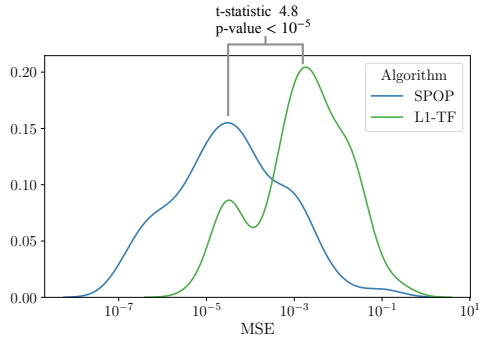


Fig. 7: Distribution of MSE on the ARMCODA data. SPOP and L1-TF are statistically different (paired t-test, level 1%).

the number of samples increases (higher sampling frequency).

d) Number of discrete states: One of the main parameters of SPOP is the number of discrete states n. A small n makes the algorithm faster but restricts the number of splines considered in the optimization. As a result, reconstruction error increases when n decreases. Time and reconstruction error are measured for $n \in \{3, 5, 7, 9, 11\}$ to evaluate this trade-off. For all experiments, $K = K^*$.

Results can be seen in Fig. 6. First, simpler signals (low $K^\star$) are better approximated. Second, more states lead to more significant execution times but lower MSE. Nevertheless, the MSE improvement quickly stalls, meaning there is no clear benefit of using more than five states.

B. Real-world data: motion capture time series

We use SPOP to compress 200 univariate time series of length 1000 from the ARMCODA data set [1]. Each signal is the coordinate of a sensor placed on a person's body. Each subject did a simple movement, e.g., hair combing, standing up, etc. One signal example is shown in Fig. 1. We report the MSE of SPOP and L1-TF using $K = 26$ segments on Fig. 7. Visually, SPOP produces a better approximation. The MSE of SPOP is also significantly better than L1-TF's (paired t-test, confidence level 1%).

V. Conclusion

The SPOP algorithm is an efficient method for univariate time series compression using continuously differentiable functions. It has a quadratic complexity. SPOP empirically outperforms baselines in terms of MSE.

We see several future directions to extend our method. First, SPOP algorithm can be adapted to multivariate time series and higher order splines. Also, when the number of segments is not fixed beforehand, we can solve a ℓ_0-penalized version of the reconstruction error as in [11]. Finally, a theoretical result on the reconstruction quality would be a valuable addition.

References

[1] S. W. Combettes, P. Boniol, A. Mazarguil, D. Wang, D. Vaquero-Ramos, M. Chauveau, L. Oudre, N. Vayatis, P.-P. Vidal, A. Roren, and M.-M. Lefèvre-Colau. Arm-coda: A data set of upper-limb human movement during routine examination. *Image Processing On Line*, 14:1–13, 1 2024.

[2] R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[3] L.J. Vostrikova. Detecting "disorder" in multidimensional random processes. 24:55–59, 1981.

[4] C. Truong, L. Oudre, and N. Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

[5] J. Bai and P. Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, 66:47–78, 1998.

[6] P. Fearnhead, R. Maidstone, and A. Letchford. Detecting changes in slope with an l0 penalty. *Journal of Computational and Graphical Statistics*, 28:265–275, 2019.

[7] V. Runge, M. Pascucci, and N. Deschamps de Boishebert. Change-in-slope optimal partitioning algorithm in a finite-size parameter space. 2020.

[8] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky. L1 trend filtering. *SIAM Review*, 51:339–360, 2009.

[9] C. R. Rojas and B. Wahlberg. How to monitor and mitigate stair-casing in l1 trend filtering. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3946–3950. IEEE, 2015.

[10] C. Wen, X. Wang, and A. Zhang. ℓ_0 Trend Filtering. *INFORMS Journal on Computing*, 35(6):1491–1510, November 2023.

[11] P. Fearnhead R. Killick and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[12] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. San, L. Tan, and Tun Tao Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105–108, February 2005.

[13] D. E. Knuth. Johann faulhaber ad sums of powers. *Mathematics of computation*, 1993.

[14] R. J. Tibshirani. Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42:285–323, 2014.