Efficient algorithms computing p-variation

Aimen Daoudi, Stéphane Junca

Efficient algorithms computing p-variation

Aimen Daoudi* †, Stéphane Junca ‡

Abstract

This work focuses on the efficient computation of p-variation. For $p > 1$, this semi-norm is associated with fractional regularity $s = 1/p < 1$ and the fractional BV space BV^s. It is well-known in probability theory and has recently gained significance in the context of hyperbolic partial differential equations. The primary motivation of this paper is to calculate p-variation efficiently for numerical schemes of hyperbolic conservation laws. The p-variation is a non-local semi-norm. Initially, based on its definition, the cost of computing the p-variation for a piece-wise constant function with N data points appears to be exponential with respect to N. However, We introduce new algorithms featuring polynomial costs, after the recent one provided by Vygantas Butkus and Rimas Norvaisa.

AMS Classification: 26A45, 6504, (35L65) .

Key words: p-variation, fractional BV spaces, conservation laws.

Contents

1 Introduction **2**

2 On p-variation **4**
2.1 Examples and properties of optimal sequences 4
2.2 Main tools to build an optimal subsequence 7

3 Algorithms to compute p-variation **16**
3.1 "Eraser" 17
3.2 "Merge" 20
3.3 "AOP"=Add One Point 22

4 Efficiency of the algorithms **24**
4.1 The cost is at least N^2 24
4.2 Cost comparisons of the three algorithms 25

A Motivations to compute the p-variation **27**

*Université Côte d'Azur, LJAD, Nice, France, aimen.daoudi@etu.univ-cotedazur.fr

†Laboratoire de Mathématiques Appliquées, Université Kasdi Merbah, Ouargla, Algerie, daoudi.aimen@univ-ouargla.dz

‡Université Côte d'Azur, Inria & CNRS, LJAD, Nice, France, stephane.junca@univ-cotedazur.fr

1 Introduction

The p-variation, $p \geq 1$, is a well-known theoretical tool in Probabilities [17] and Functional Analysis [13, 14]. The introduction of the p-variation is due to L.C. Young [35] after N. Wiener for $p = 2$ [33]. It allows functions with low regularity, typically exhibiting a fractional derivative $s = 1/p \leq 1$ [2].
The interest in effectively computing the p-variation is very recent. The first general algorithm for computing the p-variation was proposed by Vygantas Butkus and Rimas Norvaisa [6], primarily for applications in probability. This algorithm is referred to as 'Merge'. In this work, two new algorithms, 'Eraser' and 'AOP,' are presented, along with the 'Merge' algorithm applied in a slightly different context.
Let's provide an initial insight into the difficulty associated with computing the p-variation

For $p = 1$, the 1-variation, known as the total variation is the well-known semi-norm related to BV, the space of functions with bounded variation, and it is easy to compute.
For $p > 1$, there are a few rare explicit computations of the p-variation for specific functions in [5, 8]. The general case is indeed complicated.
For a finite sequence $u = (u(1), \ldots, u(N))$, $N \geq 2$ the p-variation is,

$$\textbf{p-TV}\ u := \sup_{\sigma \subset \{1,\ldots,N\}} p\text{-}\mathfrak{sum}_{\sigma}\ u, \tag{1.1}$$

$$p\text{-}\mathfrak{sum}_{\sigma}\ u = \sum_{i=1}^{|\sigma|-1} |u(\sigma(i+1)) - u(\sigma(i))|^p, \tag{1.2}$$

Where the supremum is taken over all subdivisions of $1, \ldots, N$. A subdivision σ is an ordered subset of $1, \ldots, N$. The cardinal or length of σ is denoted by $|\sigma|$.
A p-$\mathfrak{sum}$ of a sequence is the p-$\mathfrak{sum}$ involving all the terms of the sequence. The notation p-$\mathfrak{sum}\ , u$ without referring to a subdivision denotes the sum of all consecutive variations of u at the power p.

$$p\text{-}\mathfrak{sum}\ u := p\text{-}\mathfrak{sum}_{(1,\ldots,N)}\ u = \sum_{i=1}^{N-1} |u(i+1) - u(i)|^p. \tag{1.3}$$

The cost of computing a p-$\mathfrak{sum}$ of a sequence u is simply of order N. In contrast, the number of subdivisions is prohibitive, on the order of 2^N. Thus, computing a p-variation of u directly as the maximum of all p-$\mathfrak{sum}$ corresponding to all subsequences of u is too expensive. A strategy to compute **p-TV**u is to find an **optimal subdivision** $\sigma^\star$ such that $\textbf{p-TV}\ u = p\text{-}\mathfrak{sum}_{\sigma^*}\ u$. That is to find an **optimal subsequence** $u^\star = (u(i))_{i \in \sigma^\star}$ such that $\textbf{p-TV}\ u = p\text{-}\mathfrak{sum}\ u^\star$. In this paper, we propose some algorithms to find $u^\star$ efficiently.

Notice that extensive use of subsequences is made in this paper for $p > 1$. Therefore, the chosen notation for sequences is the same as that for functions. A value in a sequence is denoted $u(n)$ instead of u_n because it is more readable. Thus, for a subsequence, we prefer to use the notation $u(\sigma(k))$ instead of $u_{\sigma(k)}$ or u_{σ_k}.

In general, only for $p = 1$ this supremum is easy to compute, thanks to the triangular inequality, the whole subdivision $\sigma^* = \{1, \dots, N\}$ always yields the supremum. And the cost of this computation is simply N. That is for $p = 1$,

$$TV\,u = \textbf{p-TV}u = p\text{-}\mathfrak{sum}\ u := \underset{(1,\dots,N)}{p\text{-}\mathfrak{sum}}\ u = \sum_{i=1}^{N-1} |u\,(i+1) - u\,(i)|^p\,. \tag{1.4}$$

For $p > 1$, it is more complicated. The whole subdivision is not usually optimal to compute the supremum. For instance, let u be an increasing sequence, $u \in \mathbb{R}^{N+1}$, where $u(i) = \dfrac{i-1}{N}$, $i = 1, \dots, N+1$. In this case, if all of the $(N+1)$ points of u are taken, then, when $N \to +\infty$,

$$p\text{-}\mathfrak{sum}\ u = \sum_{i=1}^{N} |u\,(i+1) - u\,(i)|^p = N\frac{1}{N^p} = \frac{1}{N^{p-1}} \to 0.$$

But if the first term and the last one are taken, that is the subdivision $\sigma^* = \{1, N\}$, then the greatest p-$\mathfrak{sum}$ is achieved. Thus, the p-variation is,

$$\textbf{p-TV}\ u = \underset{(1,N+1)}{p\text{-}\mathfrak{sum}}\ u = |u(N+1) - u(1)|^p = |1-0|^p = 1.$$

Also, taking the first and the last values of the sequence is not always the optimal choice, for example, $u = (u(1), u(2), u(3)) = (0, 9, 0)$, here $|u(1) - u(3)|^p = 0$. But if we take all the terms of the sequence u, we get 2-TV $u = |u(1) - u(2)|^2 + |u(2) - u(3)|^2 = 162$.
As a consequence, an optimal subdivision to compute the supremum is not known by advance (except in special cases). Starting from the definition, a naive algorithm is to take the maximum of the summations for all subdivisions which has a prohibitive exponential cost.

Algorithms with a polynomial cost only appear recently [6, 11, 24]. In this paper, three algorithms are proposed "AOP", "Eraser" and "Merge". The two first are new. The third one was already found by Buktus and Norvaisa [6]. The exact cost is still not completely solved. It is at least N^2 and at most N^3. A recent preliminary numerical study [24] suggests that "AOP" is the simplest and the fastest algorithm. But there is room to improve the cost of "Eraser" and "Merge". In particular, a fourth algorithm to improve "Eraser" and "Merge", called "MEI" is proposed in [24] but not studied here.

The paper is organized as follows. In Section 2, mathematical tools are given in a self-contained way to validate mathematically all the algorithms presented. This long fundamental section is the largest part of the document Then the algorithms "Eraser", "Merge" and "AOP" are presented in Section 3. The polynomial efficiency of the algorithms are discussed in Section 4. Finally, a presentation of the motivations to compute the p-variation and the relation with fractional BV spaces are presented in the appendix.

2 On p-variation

To build efficient algorithms to compute p-variation also called the p-total variation, many useful properties and definitions are provided in this section. It is well known that a subsequence is needed to compute the p-variation of a sequence [2, 5, 6], that is to replace a supremum on a large set by a sum. Only for some special examples, the p-variation can be reached for the whole sequence in Section 2.1. These examples can be used to test algorithms computing the p-variation.

The fundamental results to build general and efficient algorithms to compute the p-variation are given in Section 2.2. This section is self-contained. Other presentations can be found in the book [5] by Michel Bruneau and in the first article presenting an efficient algorithm by Vygantas Butkus and Rimas Norvaisa [6].

2.1 Examples and properties of optimal sequences

We start with basic notions required throughout the paper.

It is said that a sequence $u = (u(i))_{1\leq i\leq N}$ has **no constant step** if

$$\forall i : 1 \leq i < N, \;\; u(i) \neq u(i+1).$$

The sequence u also only has **extreme values** if

$$\forall i : 1 < i < N \;\; (u(i) - u(i+1))(u(i) - u(i-1)) \geq 0.$$

That means that all values of u are extreme values, i.e. local maximum or local minimum:

$$u(i) \geq \max(u(i-1), u(i+1)) \qquad \text{or} \qquad u(i) \leq \min(u(i-1), u(i+1)).$$

It is convenient also to consider only strict extrema. The sequence u only has **strict extreme values** if

$$\forall i : 1 < i < N \;\; (u(i) - u(i+1))(u(i) - u(i-1)) > 0.$$

In this case the sequence cannot be constant, has no constant step and it is an oscillating sequence. An oscillating sequence is a sequence such that for all $1 < i < N$,

$$\text{sign}(u(i+1) - u(i)) = -\text{sign}(u(i) - u(i-1)).$$

The extreme values of a sequence give the first reduction of the size of a sequence to compute the p-variation ($p > 1$). Indeed, the p-variation of a sequence can be computed on the subsequence consisting of all its extreme values.

Proposition 2.1 ([2]). *Let v be the subsequence of all extreme values of u, then*

$$\textbf{p-TV}u = \textbf{p-TV}v.$$

Notice that if u has no constant step, the subsequence v containing all extreme values and with no constant step has only strict extreme values.
The result of the proposition is given in [2, 5] with a simple detailled proof in the appendix of [20].

The next proposition shows that for large p the optimal subsequence of $u = u(1), \ldots, u(n))$ to compute the p-variation is simply $(u(1), u(N)$ under the condition that the global maximum and minimum of u are only reached at the boundaries. This result is known for monotonic sequence [2, 5] but here no monotonicity is required.

Proposition 2.2. *Let $u = (u(1), \ldots, u(N))$ be a finite sequence. Assume that*

$$\forall i \text{ such that } 1 < i < N \ , \ u(1) < u(i) < u(N). \tag{2.1}$$

Then $\exists\, p_0$ such that for all $p > p_0 > 1$ there is only one optimal subdivision $\sigma^\star = \{1, N\}$ and,

$$\textbf{p-TV}u = |u(1) - u(N)|^p.$$

Proof. Let $\{\sigma(1), \sigma(2), ..., \sigma(m)\}$ a subdivision of $\{1, ..., N\}$, where $m \leq N$. We are looking to get a p_0 such that $\forall p \geq p_0$, we have:

$$\sum_{l=1}^{m-1} |u(\sigma(l)) - u(\sigma(l+1))|^p \leq |u(\sigma(N)) - u(\sigma(1))|^p,$$

we denote R as the following:

$$R = \sum_{l=1}^{m-1} \left| \frac{u(\sigma(l)) - u(\sigma(l+1))}{u(\sigma(N)) - u(\sigma(1))} \right|^p \leq 1.$$

Let be ρ,

$$\rho = \max_{\substack{i \neq j \\ 1 < i,j < N}} \left| \frac{u(i) - u(j)}{u(1) - u(N)} \right|,$$

$\rho < 1$ from assumption (2.1), therefore,

$$0 \leq R \leq \sum_{l=1}^{m-1} \rho^p = (m-1)\rho^p \leq N\rho^p \underset{p \to \infty}{\longrightarrow} 0.$$

The convergence towards 0 is uniform w.r.t. the subdivision, thus $\exists p_0$, $R < 1$ for $p \geq p_0$ and for all subdivisions. It means that $\textbf{p-TV}u[\sigma] \leq |u(1) - u(N)|^p$ for $p \geq p_0$ and therefore, $\textbf{p-TV}u = |u(1) - u(N)|^p$. □

Next, Proposition 2.3 shows that an optimal subsequence always contains the terms where the sequence reachs its maximum, its minimums, the first term and the last term of the sequence u.

Proposition 2.3. *Let $u = (u(1), \ldots, u(N))$ be a sequence with no constant step, then all indexes of the global maximum, the global minimum, and the first and the last terms of u are always in any optimal subdivision $\sigma^\star$, i.e.,*

$$\{1,N\} \cup \ arg\ max\ u\ \cup\ arg\ min\ u\ \subset \sigma^\star.$$

Proof. Let $\sigma^\star$ be an optimal subsequence. 1 Assume first that 1 and N does not belong to $\sigma\star$ then consider $\sigma = \{1\} \cup \sigma \cup \{N\}$

$$\begin{aligned} \underset{\sigma^\star}{p\text{-}\mathfrak{sum}}\ u &= \sum_{l=1}^{|\sigma^\star|-1} |u(\sigma^\star(l)) - u(\sigma^\star(l+1))|^p \\ &< \sum_{l=1}^{|\sigma^\star|-1} |u(\sigma^\star(l)) - u(\sigma^\star(l+1))|^p + |u_1 - \sigma^\star(1)|^p + |u_N - \sigma^\star(|\sigma^\star|)|^p \\ &= \underset{\sigma}{p\text{-}\mathfrak{sum}}\ u. \end{aligned}$$

Notice that the strict inequality is a consequence that u has no constant step so $|u_1 - \sigma^\star(1)|^p + |u_N - \sigma^\star(|\sigma^\star|)|^p > 0$. Then the subdivision $\sigma^\star$ is not optimal. That means that 1 and N must belong to $\sigma^\star$.
Now we prove that all indexes of the global maximum and the global minimum of the sequence u belong to $\sigma^\star$. Assume the negation. Then, there is one of the indexes of the maximum or minimum of u not belonging to the optimal subdivision $\sigma^\star$, i.e. there exists two indexes k_1, k_2 such that: $\sigma^\star(1) < k_1 < \sigma^\star(|\sigma^\star|) - 1$, $\sigma^\star(1) < k_2 < \sigma^\star(|\sigma^\star|)$,, where one of the indexes of maximum of u is between $\sigma^\star(k_1)$ and $\sigma^\star(k_1 + 1)$,(resp. one index of minimum of u is between $\sigma^\star(k_2)$ and $\sigma^\star(k_2 + 1)$). Therefore,

$$\begin{aligned} \underset{\sigma^*}{p\text{-}\mathfrak{sum}}\ u &= \sum_{l=1}^{|\sigma^\star|-1} |u(\sigma^\star(l)) - u(\sigma^\star(l+1))|^p \\ &< \sum_{l=1}^{|\sigma^\star|-1} |u(\sigma^\star(l)) - u(\sigma^\star(l+1))|^p \\ &\quad + |u_{max} - u(\sigma^\star(k_1+1))|^p + |u_{max} - u(\sigma^\star(k_1))|^p \\ &\quad + |u_{min} - u(\sigma^\star(k_1+1))|^p + |u_{min} - u(\sigma^\star(k_1))|^p. \end{aligned}$$

Thus that $\sigma^\star$ is not optimal. □

Now, the previous result is illustrated by the following oscillating example where $\sigma^\star = \{1, \ldots, N\}$. Indeed, it is the simplest way to no reducing the sequence for a sequence having no constant step. An oscillating sequences with only two values $c_1 \neq c_2$ is defined by:

$$u(i) = \begin{cases} c_1, \text{ for i even number} \\ c_2, \text{ for i odd number} \end{cases} \tag{2.2}$$

Another example used in [2, 9] where u is already an optimal sequence is now given. A sequence $u = (u(1), \ldots, u(N))$ has a decreasing amplitude means that:

$$|u(i) - u(i+1)| \leq |u(i+1) - u(i+2)| \ , \forall i : 1 \leq i \leq N-2.$$

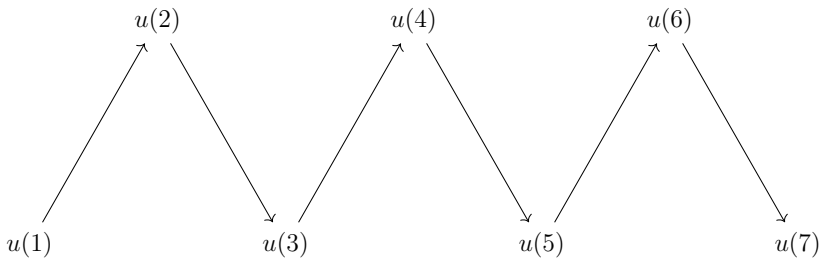

Figure 1: An oscillating sequence with only two values

Figure 2: A sequence with decreasing amplitude

The next proposition 2.4 shows that if the sequence u has a decreasing amplitude then the optimal subsequence also is the same sequence u.

Proposition 2.4 ([2]). *Let $u = (u_1, \dots, u_N)$ an oscillating sequence with no constant step that has a decreasing amplitude than, $u = u^\star$, and $\sigma^\star = \{1, \dots, N\}$.*

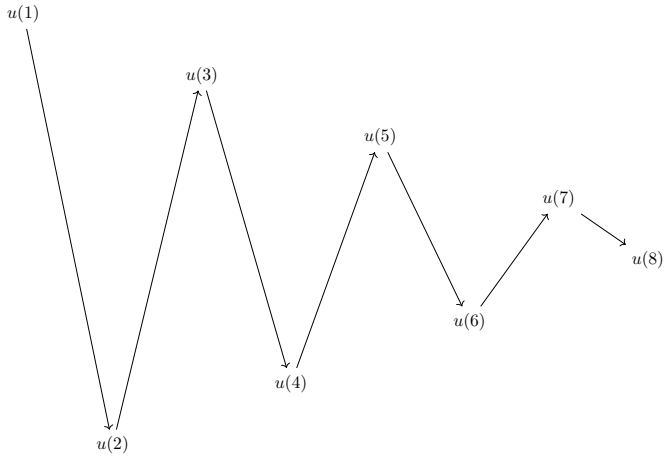

Figure 3: A sequence with decreasing amplitude

A detailled proof is in [9], Proposition 10.

2.2 Main tools to build an optimal subsequence

The main tools to build the algorithms is given in this section.
By definition of the p-variation of any finite sequence u, there exists a subsequence u^* of u such that p-$\mathfrak{sum}$ $u^* =$ **p-TV**u. How to build u^*? We proceed step by step. We start with the whole sequence. Then the following test has done on the $j+1$ consecutive terms after u_i . If $|u_{i+j} - u_i|^p \geq \sum_{l=1}^{j} |u_{i+l} - u_{i+l-1}|^p$ then the subsequence v with all the terms of u except the $j-1$ consecutive terms $u_{i+1}, \dots, u_{i+j-1}$ seems to be a smaller sequence with the same p-variation, **p-TV**$v =$ **p-TV**u. If it is the case, we can continue, continue, ... and finally expect to reach an optimal subsequence w of u such that **p-TV**$w = p$-$\mathfrak{sum}$ $w =$ **p-TV**u.

Here, we will prove that such strategy work well if we do such tests in the right order. This strategy corresponds to the natural "Eraser" algorithm presented below.
The following condition is then a fundamental condition to check,

$$H_{ij}^{<} = H_{ij}^{<}(u): \qquad |u(i+j)-u(i)|^p < \sum_{l=1}^{j} |u(i+l)-u(i+l-1)|^p, \qquad 2 \le j < N. \quad (2.3)$$

When there is no ambiguity on the sequence u considered the short notation $H_{ij}^{<}$ is used instead of $H_{ij}^{<}(u)$. In the same way we define $H_{ij}^{=}$, $H_{ij}^{\ge}$, $H_{ij}^{>}$. For instance $H_{ij}^{=}$ means,

$$H_{ij}^{=}: \qquad |u(i+j)-u(i)|^p = \sum_{l=1}^{j} |u(i+l)-u(i+l-1)|^p. \quad (2.4)$$

We will see below that if $H_{ij}^{<}$ is true for all i and all j then u is already and optimal sequence to compute its p-variation. To shorthen the notations,

$H_{\cdot j}^{\le}$ means that all inequalities $H_{ij}^{\le}$, $i = 1, \dots, N-j$ are satisfied where $2 \le j < N$.

This fundamental proposition characterizes optimal sequences.

Proposition 2.5 (Optimal sequence to compute p-TV).
Let $p > 1$ and $u = (u(1), \dots, u(N))$.
p-TV$u = p$-$\mathfrak{sum}$ u *if and only if all inequalities $H_{\cdot j}^{\le}(u)$ are true, $j = 2, \dots, N-1$.*

In practice, if all strict inequalities $H_{ij}^{<}(u)$ are true, u is an optimal sequence to compute **p-TV**u. This last criteria is preferred because it allows to reduce the size of the subsequence. Less terms has a sequence, less expansive is the cost of computing a p-$\mathfrak{sum}$. Let us explain on a simple example the reduction of the size of the sequence on a typical example. Let $p = 2$ and u be the sequence $u = (0, 2a, a, 3a)$ with $a > 0$ has exactly two optimal subsequences to compute **p-TV**u, $u = (u(1), u(2), u(3), u(4))$ and $v = (u(1), u(4)) = (0, 3a)$ since,

$$\begin{aligned} \textbf{p-TV}u = 9a^2 &= p\text{-}\mathfrak{sum}\ v = |0-3a|^2 = 3^2a^2 \\ &= p\text{-}\mathfrak{sum}\ u = |0-2a|^2 + |2a-a|^2 + |a-3a|^2 = 2^2a^2 + a^2 + 2^2a^2. \end{aligned}$$

Proof. By definition of **p-TV**u, p-$\mathfrak{sum}$ $u \le$ **p-TV**u is always true. No consider v a subsequence of u associated the a subdivision σ of length k, $v = (u(\sigma(1), \dots, u(\sigma(k))$. Using inequalities $H_{ij}^{\le}(u)$ yield

$$\begin{aligned} p\text{-}\mathfrak{sum}\ v &= \sum_{i=2}^{k} |v(i)-v(i-1)|^p = \sum_{i=2}^{k} |u(\sigma(i))-u(\sigma(i-1))|^p \\ &\le \sum_{i=2}^{k} \sum_{j=\sigma(i-1)}^{\sigma(i)-1} |u(j+1)-u(j)|^p = \sum_{j=\sigma(1)}^{\sigma(k)} |u(j)-u(j-1)|^p \le p\text{-}\mathfrak{sum}\ u. \end{aligned}$$

Thus p-$\mathfrak{sum}$ $u =$ **p-TV**u.

Conversely, if one inequality $H_{ij}^{\leqslant}$ is not satisfied for a couple (i,j) such that $1 \leq i \leq N-2$, $2 \leq j \leq N-i$, then $H_{ij}^{>}$ is true. Consider $v = (u(1), \ldots, u(i), u(i+j), \ldots, u(N))$ the subsequence of u without the $j-1$ terms $u(i+1), \cdots, u(i+j-1)$, then

$$\begin{aligned} p\text{-}\mathfrak{sum}\ u &= \sum_{l=1}^{i-1} |u(l+1)-u(l)|^p + \sum_{l=i}^{i+j-1} |u(l+1)-u(l)|^p + \sum_{l=i+j}^{N1} |u(l+1)-u(l)|^p \\ &< \sum_{l=1}^{i-1} |u(l+1)-u(l)|^p + |u(i+j)-u(i)|^p + \sum_{l=i+j}^{N1} |u(l+1)-u(l)|^p = p\text{-}\mathfrak{sum}\ v. \end{aligned}$$

Thus, u is not an optimal sequence and the proposition is proven. □

The goal of this paper is to look for a subsequence v of u and compute **p-TV**u with the p-$\mathfrak{sum}$ v. That is transforming a problem with, at a first sight, an exponential cost, by a p-$\mathfrak{sum}$ which has a linear cost. The main idea is to use check inequality $H_{ij}^{\leqslant}$ and when it is false to erase all the terms between $u(i)$ and $u(i+j)$ to build a subsequence v without loosing **p-TV**u. The subsequence has to satisfies **p-TV**v = **p-TV**u. When the subsequence satisfies Proposition 2.5, that is v is an optimal sequence to compute **p-TV**v with p-$\mathfrak{sum}$ v, then p-$\mathfrak{sum}$ v = **p-TV**v = **p-TV**u. If v does not satifies the proposition then continue the extraction of a subsequence, etc ...
To use such a strategy, we have to be careful. Starting the test of inequlity $H_{ij}^{\leqslant}$ on long length of consecutive terms, i.e. large j, and finishing by small length can be a very bad idea. For instance, consider the sequence $u = (1,0,1,2,3)$ with 5 terms. The condition $H_{14}^{\leqslant}$ is false. If the sequence $v = (u(1), u(6)) = (1,4)$ is extracted then

$$p\text{-}\mathfrak{sum}\ v = 9 < (1-0)^2 + (3-0)^2 = \textbf{p-TV}u = 10.$$

The next proposition overcomes this possible difficulty. Indeed, Proposition 2.6 gives sufficient conditions to have only one subsequence to compute the p-variation when inequality $H_{1(N-1)}^{\leqslant}$ is false

Proposition 2.6. [When $H_{1(N-1)}^{\geqslant}$ yields the p-variation]
Let $p > 1$, $u = (u(1), \ldots, u(N))$, $u^* = (u(1), u(N))$ and $N \geq 3$.
If the inequality $H_{1(N-1)}^{<}$ is false but for all $j < N-1$ the inequalities $H_{.j}^{\leqslant}$ are true then,

(P1) The sequence u has no constant step and $u(1) \neq u(N)$.

(P2) u^* is an optimal subsequence of u, **p-TV**$u = p$-$\mathfrak{sum}\ u^* = |u(1)-u(N)|^p > 0$.

(P3) If $H_{1(N-1)}^{>}$ is satisfied, then, u^* is the unique optimal subsequence to compute **p-TV**u,

$$\textbf{p-TV}u = p\text{-}\mathfrak{sum}\ u^*.$$

(P4) If the equality $H_{1(N-1)}^{=}$ is true then there are exactly two optimal subsequences, the shortest u^* and the longest u,

$$\textbf{p-TV}u = p\text{-}\mathfrak{sum}\ u^* = p\text{-}\mathfrak{sum}\ u.$$

(P5) $\{u(1), u(N)\} = \{\min u, \max u\}$ *and* $\min u < u(i) < \max u$, *where* $1 < i < N$.

Notice that, in general, if u has pnly one constant step there is maybe no uniqueness of an optimal subsequence to compute **p-TV**. Fo instance $u = (u(1), u(2), u(3)) = (1, 2, 2)$ has three subsequences to compute the p-variation
$(u(1), u(2))$, $(u(1), u(3))$and $u = (u(1), u(2), u(3))$.

Proof. The inequality $H^{<}_{1(N-1)}$ is false means $H^{>}_{1(N-1)}$ or $H^{=}_{1(N-1)}$is true, Now, all properties can be proved.

For (P1): The conditions $H^{<}_{i2}$ are true for all $i \leq N-2$ means

$$0 \leq |u(i+2) - u(i)|^p < |u(i+1) - u(i)|^p + |u(i+2) - u(i+l)|^p.$$

The inequality is strict so $|u(i+1) - u(i)| > 0$ or $|u(i+2) - u(i+1)| > 0$. But, if one term is zero there is an inequality. Thus, the two terms are positive. This property is true for all $1 \leq i \leq k-2$, so u has no constant step.

Hence, all the terms on the right hand side of inequality $H^{>}_{1(N-1)}$ are positive and $|u(1) - u(N)| > 0$, i.e. $u(1) \neq u(N)$.

For (P2): Let is v a subsequence of u. If $v = u^*$ then p-$\mathfrak{sum}$ $v = |u(1) - u(k)|^p =$ **p-TV**u^*.

If $v = u$ then p-$\mathfrak{sum}$ $v = p$-$\mathfrak{sum}$ $u <$ **p-TV**u^* by inequality $H^{>}_{1(N-1)}$.

Now, in all the sequel, only the case $u \neq v \neq u^*$ is considered. So, some terms of the sequence u are missing in v. That means that there exists i and $j < k-1$ such that the terms $u(i)$, $u(i+j)$ are in v but not $u(i+l)$ for all $0 < l < j$. However, inequality $H^{<}_{ij}$ is true since $j < k-1$. Now, consider the sequence w containing all the v terms and filling the gap between $u(i)$, $u(i+j)$ by all the terms $u(i+l)$ for all $l = 1, \ldots, j-1$. So u is a subsequence of w, w is a subsequence of u and,

$$(p\text{-}\mathfrak{sum}\ w) - (p\text{-}\mathfrak{sum}\ v) = \sum_{l=1}^{j-1} |u(i+l) - u(i+l+1)|^p - |u(i+j) - u(i)|^p > 0$$

since inequality $H^{<}_{ij}$ is true. That means p-$\mathfrak{sum}$ $v < p$-$\mathfrak{sum}$ w. Continuing to fill all the gaps of v in the same way and using the fact that all corresponding $H^{<}_{ij}$ are true finally yields,

$$p\text{-}\mathfrak{sum}\ v < p\text{-}\mathfrak{sum}\ u. \tag{2.5}$$

The last inequality is so important that we rewrite the proof by a direct computation. If $v \neq u^*$ and $v \neq u$ then, there exists a subdivision σ with length $L < k$ such that $v = (u(\sigma(i)), i = 1, \ldots, L)$ and,

$$\begin{aligned} p\text{-}\mathfrak{sum}\ v &= \sum_{i=2}^{L} |u(\sigma(i) - u(\sigma(i-1)|^p \\ &< \sum_{1<i\leq L} \ \sum_{\sigma(i-1)<l\leq\sigma(i)} |u(l) - u(l-1)|^p = p\text{-}\mathfrak{sum}\ u. \end{aligned}$$

Notice that at the second line that the inequality is strict because, v has at least one term less than u because $v \neq u$, and at most $k-3$ consecutive terms less than u because $v \neq u^*$. The corresponding condition $H_{ij}^{<}$ for these missing consecutive terms is true. Thus, Inequality (2.5) is proved.

Now, we can conclude from the inequality (2.5). Since H_k^1 is false, $p\text{-}\mathfrak{sum}\ u \leq |u(1)-u(N)|^p$ and for all strict subsequences v of u we have $p\text{-}\mathfrak{sum}\ v < p\text{-}\mathfrak{sum}\ u \leq |u(1)-u(N)|^p$.

Finally for all subsequence v of u, $p\text{-}\mathfrak{sum}\ v \leq |u(1)-u(N)|^p$, that means that $\mathbf{p\text{-}TV}u = |u(1)-u(N)|^p$.

For (P3): This is a consequence of the proof of (P2).

For (P4): In case of equality there are two subsequences, one is u with the maximal length and the other one u^* with the minimal length. Notice that the equality is always possible. For $p=2$ it suffices to take this minimal example [2], $u=(0,a,a-b,2a+b)$ with $0<b<a$ and $b=a/2$. For $1<p\neq 2$, the existence of b is obtained by the intermediate value theorem. The function $g(b)=a^p+b^p+a^p-(2a-b)^p = p\text{-}\mathfrak{sum}\ u - p\text{-}\mathfrak{sum}\ u^*$ is continuous, $g(0)=(2-2^p)a^p<0$ and $g(a)=2a^p>0$ so the existence of a positive $b<a$ follows.

For (P5): by the proposition (2.3), we can get that the indexes of maximum and minimum of $(u_n)_{1\leq n\leq k}$ are in $\sigma^\star=(1,N)$, since its length is 2 and by (P1) $u(1)\neq u(N)$. Moreover, that means that the extrema of the sequence u are only reached at the boundary $\{1,N\}$, thus (P3) is proved.

□

The next result generalizes Proposition 2.6 and it will be used later to validate all algorithms presented in this paper

Theorem 2.1 (**subsequence containing an optimal one**).
Let $p>1$ and $u=(u(1),\ldots,u(N))$, where $N\geq 3$, let $k\leq N$.
If $H_{.j}^{<}$, $2\leq j<k$, $H_{1k}^{<}, H_{2k}^{<}, ..., H_{(\alpha-1)k}^{<}$ are true, and $H_{\alpha k}^{<}$ is false. then, there exists an optimal subsequence u^* and its optimal subdivision σ^* to compute $\mathbf{p\text{-}TV}u$ such that,

$$u^* \subset (u(1),\ldots,u(\alpha),u(\alpha+k),\ldots,u(N)), \qquad \sigma^* \cap \{\alpha+1,...,\alpha+k-1\} = \emptyset.$$

Proof. Proposition 2.6 yields that the extrema of u on the interval $\alpha\leq l\leq \alpha+k$ are only reached for the terms $u(\alpha)$ and $u(\alpha+k)$. To fix the notations assume that

1. $u(\alpha) = \min\limits_{\alpha\leq l\leq\alpha+k} u(l)$.
2. $u(\alpha+k) = \max\limits_{\alpha\leq l\leq\alpha+k} u(l)$.
3. $u(\alpha)<u(i)<u(\alpha+k)$ for all $\alpha<i<\alpha+k$.

If it is not the case, it suffices to multiply the sequence -1.
Now, for any subdivision $\sigma \subset \{1, ..., N\}$ a suitable subdivision τ is built such that,

$$\tau \cap \{\alpha + 1, ..., \alpha + k - 1\} = \emptyset \quad \text{and} \quad p\text{-}\underset{\sigma}{\mathfrak{sum}}\ u \leq p\text{-}\underset{\tau}{\mathfrak{sum}}\ u.$$

For this purpose, let $\tilde{\sigma}$ a subdivision with cardinal ν defined by,

$$\tilde{\sigma} = \sigma \cap \{\alpha + 1, ..., \alpha + k - 1\}.$$

Depending on ν there are 3 cases to consider, $\nu = 1$, $\nu = 2$, and $\nu \geq 3$.

First, when $\nu = 1$, $\tilde{\sigma}$ has only one element called δ, $\tilde{\sigma} = \{\delta\}$.
Let β be the largest index in σ less than α, and γ be the smallest in σ greater than $\alpha + k$,

$$\beta < \alpha < \alpha + k < \gamma.$$

Now, they are only four possibilities illustrated in Figures [4,5,6,7]. The subdivision τ is built for each case:

case 1 $u(\beta) \geq u(\alpha)$ and $u(\alpha + k) \geq u(\gamma)$. $\tau = (\sigma - \{\delta\}) \cup \{\alpha, \alpha + k\}$.

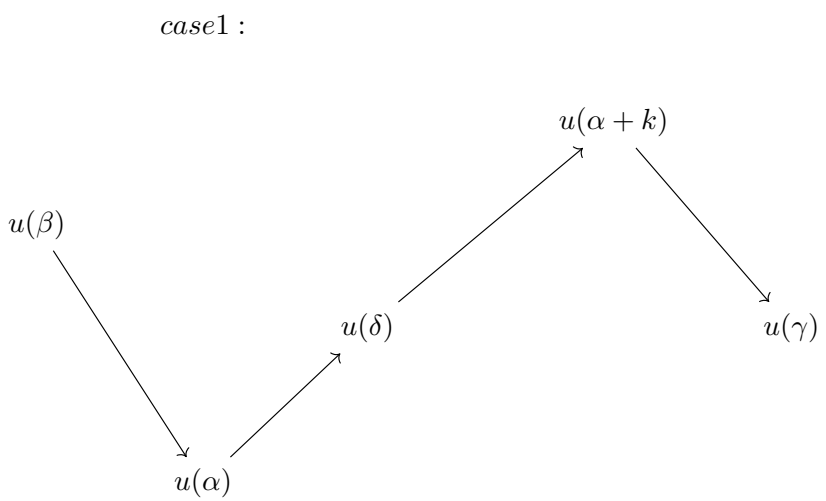

Figure 4: $\nu = 1$, case 1.

case 2 $u(\beta) < u(\alpha)$ and $u(\alpha + k) < u(\gamma)$. $\tau = \sigma - \{\delta\}$

case 3 $u(\beta) < u(\alpha)$ and $u(\alpha + k) \geq u(\gamma)$. $\tau = (\sigma - \{\delta\}) \cup \{\alpha + k\}$

case 4 $u(\beta) \geq u(\alpha)$ and $u(\alpha + k) < u(\gamma)$. $\tau = (\sigma - \{\delta\}) \cup \{\alpha\}$

$case2$:

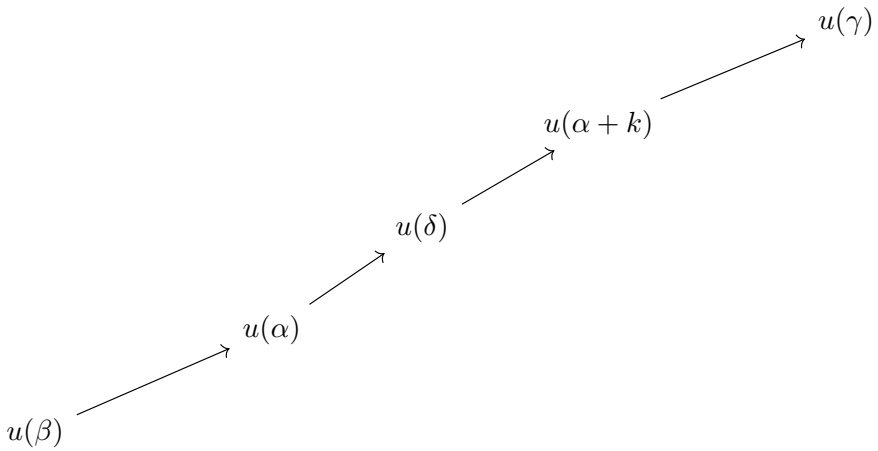

Figure 5: $\nu = 1$, four case 2.

Second, when $\nu = 2$, i.e. $\tilde{\sigma} = \{\delta, \epsilon\}$. Let $\sigma^- = \sigma - \tilde{\sigma}$.

Since the sequence $\{\alpha, \delta, \epsilon, \alpha + k\}$ has 4 terms, there are more cases. Indeed 8 where τ is defined as follows,

case 1.1 $u(\beta) \geq u(\alpha)$, $u(\alpha + k) \geq u(\gamma)$ and $u(\delta) < u(\epsilon)$ $\tau = \sigma^- \cup \{\alpha, \alpha + k\}$

case 1.2 $u(\beta) \geq u(\alpha)$, $u(\alpha + k) \geq u(\gamma)$. and $u(\delta) > u(\epsilon)$ $\tau = \sigma^- \cup \{\alpha, \alpha + k\}$

case 2.1 $u(\beta) < u(\alpha)$, $u(\alpha + k) < u(\gamma)$ and $u(\delta) < u(\epsilon)$ $\tau = \sigma^-$

case 2.2 $u(\beta) < u(\alpha)$,$u(\alpha + k) < u(\gamma)$ and $u(\delta) > u(\epsilon)$ $\tau = \sigma^-$

case 3.1 $u(\beta) < u(\alpha)$, $u(\alpha + k) \geq u(\gamma)$ and $u(\delta) < u(\epsilon)$ $\tau = \sigma^- \cup \{\alpha + k\}$

case 3.2 $u(\beta) < u(\alpha)$, $u(\alpha + k) \geq u(\gamma)$ and $u(\delta) > u(\epsilon)$, $\tau = \sigma^- \cup \{\alpha + k\}$

case 4.1 $u(\beta) \geq u(\alpha)$,$u(\alpha + k) < u(\gamma)$ and $u(\delta) < u(\epsilon)$ $\tau = \sigma^- \cup \{\alpha\}$

case 4.2 $u(\beta) \geq u(\alpha)$, $u(\alpha + k) < u(\gamma)$ and $u(\delta) > u(\epsilon)$ $\tau = \sigma^- \cup \{\alpha\}$

Third and final case $\nu \geq 3$, $\tilde{\sigma} = \{\delta(1), ..., \delta(\nu)\}$.

For the sequence $(u(l))_{\delta(1) \leq l \leq \delta(M)}$ which is u restricted to $\tilde{\sigma}$, the inequalities $H^{<}_{\delta(1)j}$ are true, where $j = \delta(\nu) - \delta(1)$, thus,

$$\begin{aligned} |u(\delta(\nu)) - u(\delta(1))| &< \sum_{i=\delta(1)}^{\delta(\nu-1)} |u(\sigma(\delta(i))) - u(\sigma(\delta(i+1)))|^p \\ &< \sum_{i=\alpha}^{\alpha+k-1} |u(\sigma(\delta(i))) - u(\sigma(\delta(i+1)))|^p \leq |u(\alpha) - u(\alpha + k)|^p . \end{aligned}$$

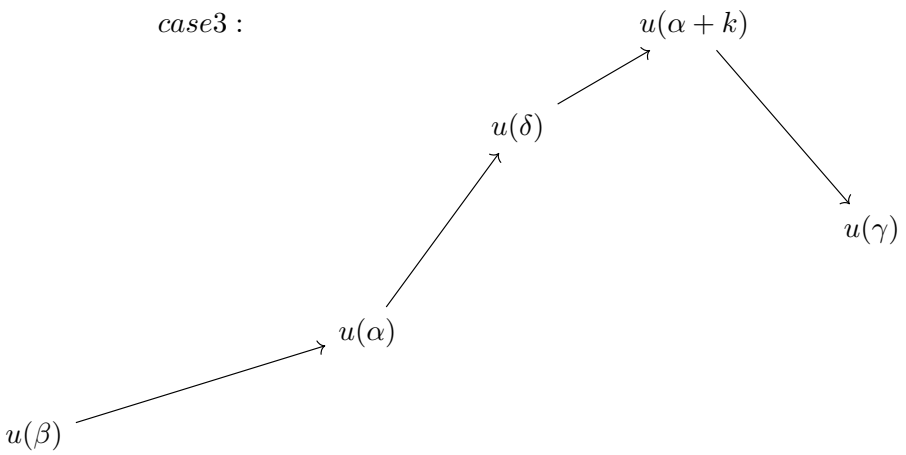

Figure 6: $\nu = 1$, four case 3.

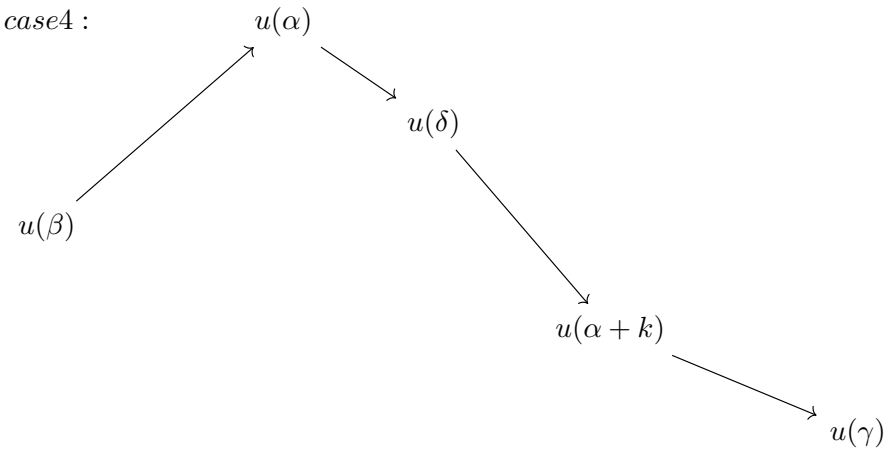

Figure 7: $\nu = 1$, four case 4.

Let $\sigma^- = \sigma - \{\delta(2), \ldots, \delta(\nu - 1)\}$ and build τ as when $\nu = 2$, completing σ^- or not with $\{\alpha, \alpha + k\}$, according to the previous eight cases. □

Next Proposition 2.7 shows that inequalities $H^<_{\bullet j}$, where j is odd are consequence of the same inequalities for smaller j when the sequence has no constant step.

Proposition 2.7 (Only even number of consecutive terms to check).
Let $u = (u(1), \ldots, u(N))$ be a sequence that has only local extrema and non constant step. If all inequalities $H^<_{\bullet j}$ are true for all $2 < j \leq 2k - 1$, then, $H^<_{\bullet 2k}$ is true.

The number of consecutive terms in the inequality $H^<_{ij}$ is $j + 1$. The proposition means that the $H^<_{ij}$ has to be checked for $j + 1$ even so j odd. Notice that $H^<_{\bullet 2}$ is true since the sequence has only local extrema and nonconstant step.

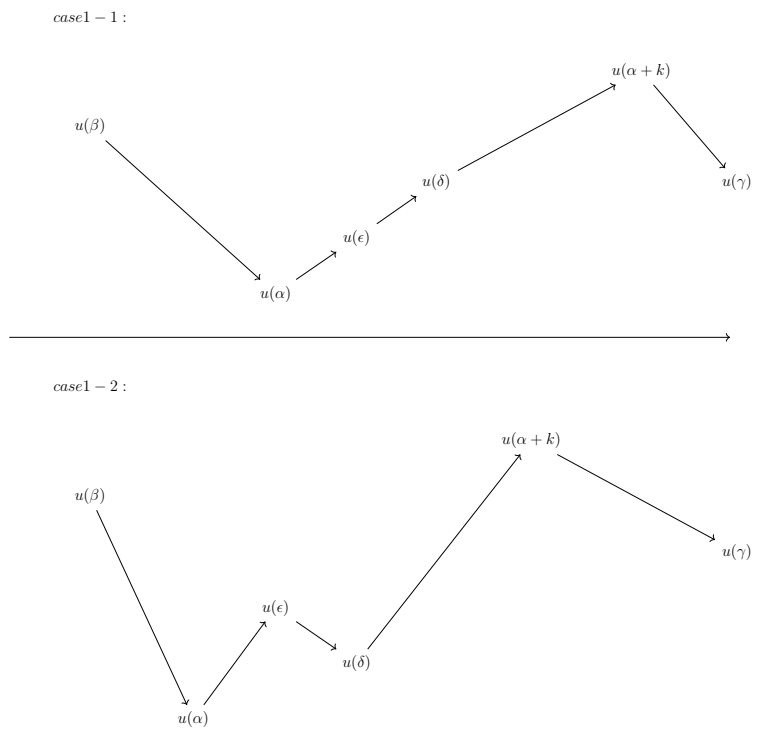

Figure 8: $\nu = 2$, case 1.1 and 1.2 .

Proof. Assume that inequalities $H^{<}_{.j}$ are true for all $2 < j < 2k$. Let i be fixed such that $i + 2k \leq N$. The goal is to prove that $H^{<}_{i,2k}$ is true.
Since the sequence u has only extrema and j is even, thus,

$$\begin{aligned} u(i) < u(i+1), \quad u(i+1) > u(i+2), \quad \cdots, \\ \cdots, \quad u(i+2k-2) < u(i+2k-1), \quad u(i+2k-1) > u(i+2k). \end{aligned}$$

If the above inequalities are in the reverse order, it suffices to multiply the sequence by (-1). They are two cases.

1. if $u(i) \geq u(i+2k)$ then $u(i+1) > u(i+2k)$ and $|u(i)-u(i+2k)| < |u(i+1)-u(i+2k)|$. Now, using inequality $H^{<}_{i+1,2k-1}$ yields

$$\begin{aligned} |u(i) - u(i+2k)|^p < |u(i+1) - u(i+2k)|^p &< \sum_{l=1}^{2k-1} |u(i+l) - u(i+l+1)|^p \\ &< \sum_{l=0}^{2k-1} |u(i+l) - u(i+l+1)|^p. \end{aligned}$$

   That means that $H^{<}_{i,2k}$ is true.

2. if $u(i) < u(i+2k)$ then $|u(i) - u(i+2k-1)| > |u(i) - u(i+2k)|$ since $u(i) <$

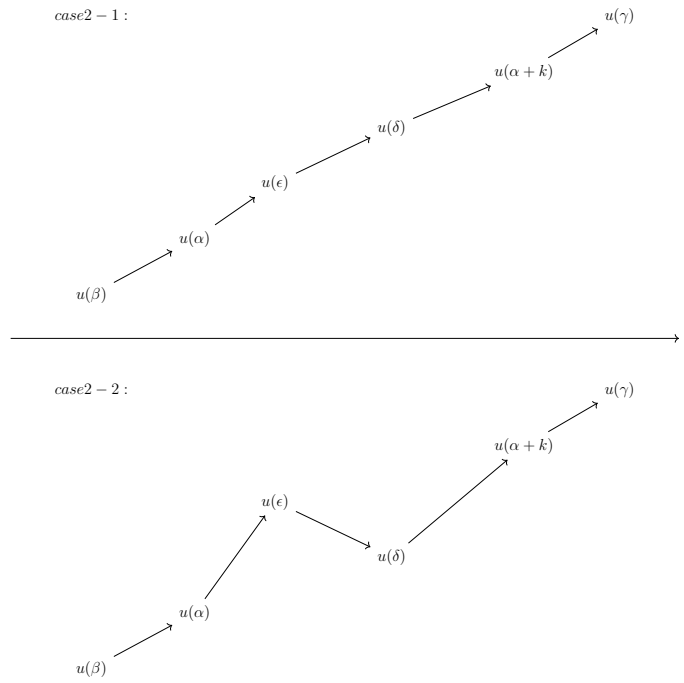

Figure 9: $\nu = 2$, case 2.1 and 2.2 .

$u(i+2k) < u(i+2k-1)$. Now, using inequality $H^{<}_{i,2k-1}$ yields

$$|u(i) - u(i+2k)|^p < |u(i) - u(i+2k-1)|^p < \sum_{l=0}^{2k-2} |u(i+l) - u(i+l+1)|^p$$
$$< \sum_{l=0}^{2k-1} |u(i+l) - u(i+l+1)|^p.$$

That means again that $H^{<}_{i,2k}$ is true.

Therefore, the condition $H^{<}_{i2k}$ is always true for all i and $H^{<}_{\bullet 2k}$ also. □

All of these properties are used to build polynomial algorithms in section 3. The point is to construct many subsequences before computing the p-variation of an initial sequence as the p-$\mathfrak{sum}$ of the final subsequence built.

3 Algorithms to compute p-variation

In this section, algorithms are presented to compute the p-TV semi-norm of a sequence $u = (u(1), ..., u(N))$ and $p > 1$. Starting with the definition of **p-TV**u, a first naive algorithm is the following.

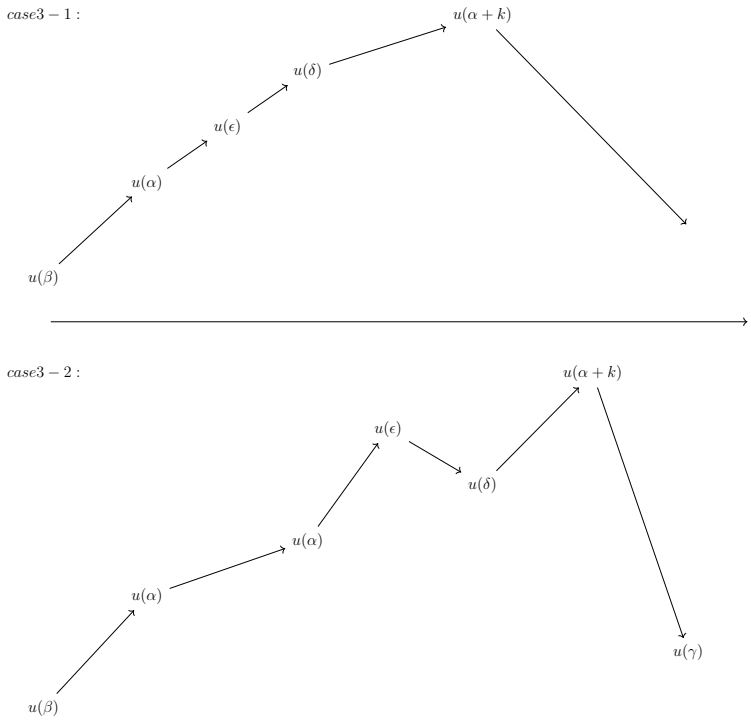

Figure 10: $\nu = 2$, case 3.1 and 3.2 .

Algorithm 0: "Exhaustif"

For all $\sigma \in \{1, ..., N\}$, compute $p\text{-}\mathfrak{sum}_{\sigma}\ u$ and take the supremum of all $p\text{-}\mathfrak{sum}$.

This algorithm is implemented in [24]. It is too costly. The number of subdivisions is of order 2^N thus the cost is at least 2^N. Moreover, the computation of a $p\text{-}\mathfrak{sum}$ is of order N thus, the algorithm "Exhaustif" is at most of order $N2^N$. It can only be used when the number N of data is small. It is the reason why other algorithms are proposed in this paper.

3.1 "Eraser"

The principle of "Eraser" is to build a subsequence of u called u^* to compute **p-TV**u with $p\text{-}\mathfrak{sum}\ u^*$. Thus, transforming a problem with an exponential cost to a problem with a linear cost. Computating a $p\text{-}\mathfrak{sum}$ has a cost of order of the length of the sequence. For this purpose, we know that all inequalities $H_{ij}^{<}$ have to be fullfilled to obtain an optimal subsequence, Proposition 2.5. If it is not the case for an inequality, as in Proposition 2.6, we can extract a subsequence. Indeed, if inequality $H_{ij}^{<}$ is wrong, we can extract a subsequence v of u by erasing all the $(j-1)$ terms between $u(i)$ and $u(i+j)$ and keep **p-TV**u = **p-TV**v if all inequalities $H_{ik}^{<}$ are true for $k < j$, Theorem 2.1. How to implement such algorithm? It suffices to start with $j = 2$, testing the validity of the inequality $H_{ij}^{<}$ for $i = 1, \ldots, N-j$. If the equalities are true, we continue with $j = 3$ and so on, increasing j by one if all inequalities $H_{ij}^{<}$ are true for all i. Using Proposition 2.7, j can be increased by 2, to consider only j odd. Thus, the sequence of j is $2, 3, 5, 7, 9, \ldots$. Only

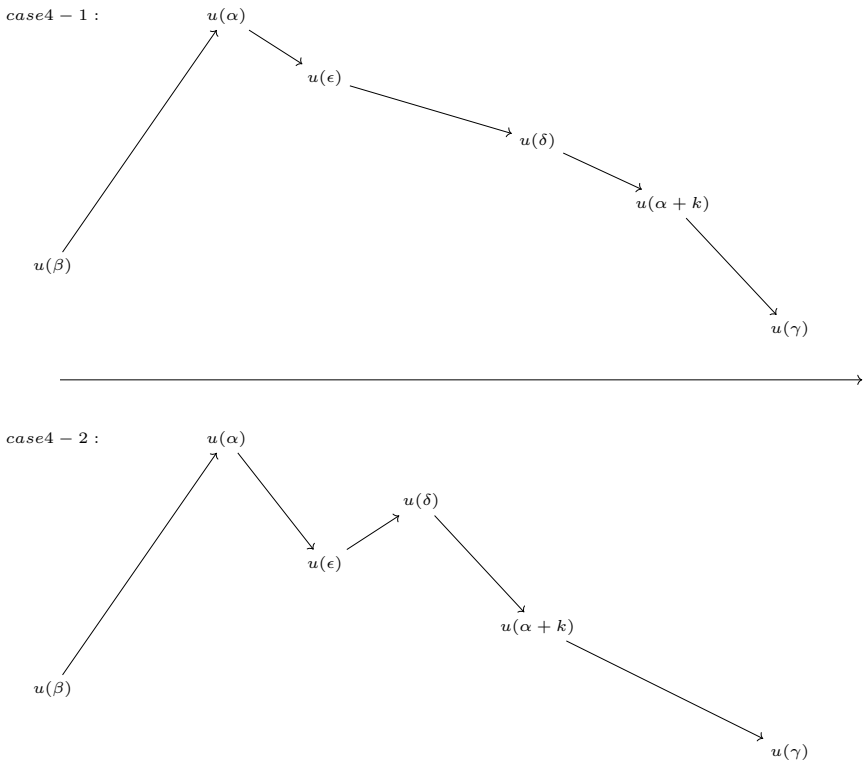

Figure 11: $\nu = 2$, case 4.1 and 4.2.

the first value of j is even to extract from the sequence u, the subsequence containing all the extreme values of u (local maximum and local minimum). Finally, if we reach $j = N-1$, there is only one inequality to test. If is true, the initial sequence is an optimal sequence and p-$\mathfrak{sum}$ $u =$ **p-TV**u.
A problem occurs if, during this process, one inequality $H^{<}_{ij}(u)$ is not true. We can erase the terms between $u(i)$ and $u(i+j)$, obtaining a subsequence v which has the same p-variation as u. Notice that inequality $H^{<}_{i2}(v)$ can be false, because it involves $v(i) = u(i), v(i+1) = u(i+j), v(i+2) = u(i+j+1)$. Thus, to apply Theorem 2.1, we have to check all inequalities $H^{<}_{kq}$ with all $q \leq j$, k such that the subsequence $(v(k), v(k+1), \ldots, v(k+q))$ involves $v(i)$ and $v(i+1)$. If all such inequalities are true, we can continue the process for $k = i+1$ and $q = j$.
A simple way to check these inequalities for v is to restart the process at the beginning with the sequence v instead of the sequence u, that is $H^{<}_{12}(v)$, $H^{<}_{22}(v)$, $H^{<}_{32}(v)$, and so on.
In the following algorithm, "com:" means commentary.

Algorithm 1a: "Eraser"

Eraser.0(u,p)

com: Input: sequence $u, p > 1$

1. u:= all extreme points of the sequence u

2. N:= size(u)

3. **For** $j = 3$ **to** $N-1$, j odd, **do**

4. **For** $i = 1$ **to** $(N-j)$ **do**

5. **If** $H_{ij}^{<}$ is false **then**

com: "Erase" $u(i+1), \ldots, u(i+j-1)$, $j-1$ terms of u to extract a subsequence:

6. u:= the subsequence of i without the j-1 terms between $u(i)$ and $u(i+j)$

com: Break, restart at the beginning,

7. Goto line 1.

8. **End If**

9. **End do**

10. **End do**

11. Result: u

The programm certainly finishes because there is only a finite number of tests and the size of the subsequence decreases after each extraction and yields to a final subsequence z. Moreover, at the end, all inequalitoes $H_{ij}^{<}(z)$ are fulfilled, thus
p-TV$u =$ **p-TV**$z = p$-$\mathfrak{sum}$ z. The first version of "Eraser", "Eraser.0" can be improved.

After one erasing, when $H_{ij}^{<}(u)$ is false, the subsequence $v = (u(1), \ldots, u(i), u(i+1), u(N))$ has the size $N = N-(j-1)$. Only the inequalities $H_{kq}^{<}$, $q < j$ or $q = j$ and $k+j = i+1$, involving $v(i)$ and $v(i+1)$ has to bo done. The other one for too small k or too big k have already been tested in the sequence u. That means for $q < j$, $k \leq i \leq i+1 \leq k+q$, that is k beetween $i+1-q$ and i. These local tests containing the terms $u(i)$ and $u(i+j) = v(i+1)$ are less costly than restarting the program with the sequence v instead of the sequence u.
If all the tested inequalities $H_{kq}^{<}$, $q < j$ are valid, then the initial process is continued with $k = i+1-j$ and the inequality $H_{kj}^{<}$.
Else, an extraction occurs for a fixed k_1 and $q < j$, and local tests has to be done around $v(k_1)$ and $v(k_1+1)$ that is doing the local tests with $i = k_1$.
Algorithm 1b: "Eraser" improved

Eraser.1(u,p)

1. Extract only extreme points from the sequence u and N=size(u)

2. **For** $j = 3$ **to** $N-1$, j odd , **do**

3. $i := 1$

4. **Repeat**

5. **If** $H_{ij}^{<}$ is false **Then** "erase" the j-1 terms between $u(i)$ and $u(i+j)$
6. **For** $q = 2$ **to** $j-1$, $q = 2$ or q odd,
7. **For** $k = i+1-q$ **to** i **do**,
8. **If** $H_{kq}^{<}$ is false **then** "erase"; i:=k and go to line 6
9. **End If**
10. **End do**
11. **End do**
12. i:= i +1 -j ; Goto line 4.
13. **Else** i=i+1
14. **End If**
15. **Until** $i > (N-j)$
16. **End do**
17. Result: u

.

3.2 "Merge"

The first efficient algorithm to compute p-variation was built by Vygantas Buktus and Rimas Norvaisa [6] for applications in Probabilities which are based on the famous algorithm "divide and conquer". Recently other algorithms are proposed in [11, 24]. A natural question concerns the initialization of such an algorithm to reduce the number of merging. Recently, an interesting idea was proposed in [24] to improve Merge and Eraser with the algorithm "MEI" explained at the end of this subsection.

The principle of "Merge" is as follows. Let $u = (u(1), \ldots, u(n))$ and $v = (v(1), \ldots, v(m))$ two sequences. If u and v are two optimal sequences, i.e. $\mathbf{p\text{-}TV}u = p\text{-}\mathfrak{sum}\ u$ and $\mathbf{p\text{-}TV}v = p\text{-}\mathfrak{sum}\ v$, then, the concatenation of the two sequences $L = (u, v)$ is "almost" an optimal sequence with size $N = n + m$. This is almost true because already many inequalities are true $H_{ij}^{<}$. For instance, all inequalities $H_{\cdot 2}^{<}$ are true, except at most for the subsequences,

$$(L(n-1), L(n), L(n+1)) = (u(n-1), u(n), v(1)),$$
$$(L(n), L(n+1), L(n+2)) = (u(n), v(1), v(2)).$$

That means there are only 2 inequalities to check to satisfy $H_{.2}^{<}$. If an inequality is not satisfied, then we proceed locally as for eraser, erasing one term and then on the new restart the process to check on the new concatenated chain, without $u(n)$ or $v(1)$ the new two cases to check. When the process is stabilized for $j = 2$, we continue with $j = 3$ and so on. Finally at the end, we obtain a sub-sequence of L which is optimal to compute **p-TV**L. It turns to use this mechanism to build an optimal sequence. Precisely, inequalities $H_{ij}^{<}$ tested have to contain some values of u and v, $1 \leq i \leq n \leq n+1 \leq i+j \leq N$, i.e. $\max(1, n+1-j) \leq i \leq \min(n, N-j)$.
The following algorithm merges two optimal sequences in a new optimal one.

Algorithm 2a: Conquer

Conquer(u,v,p)

com: u and v are optimal sub-sequences.

1. $L := (u, v)$; n:=size(u); m := size(v); N:=size(L).
2. **For** j= 2 to N-1, j =2 or j odd do
3. **For** i= max(1,n+1-j) to max(n,N-j) do
4. **If** $H_{ij}^{\geq}$ **Then** $u := (L(1), \ldots, L(i))$, $v := (L(i+j), \ldots, L(N))$; Goto line 1
5. **End If**
6. **End For**
7. **End For**
8. Result:L

The initialization of "Merge" is as follows to compute **p-TV**u with $u = (u(1), \ldots, u(N))$ We assume that the sequence u has no constant step and to simplify the exposition, that N is a power of 2, $N = 2^{K+1}$, $K \geq 1$. The initial sequence is divided in $N/2$ sequences $L[i] = (u(2i-1), u(2i)$, $i = 1, \ldots, N/2$. These 2^K sequences are optimal since u has no constant step. Now, $L[1]$ and $L[2]$ are merged in an optimal sequence with the algorithm "Conquer". The same is done until $L[2^K - 1]$ and $L[2^K]$. Now, there are 2^{K-1} optimal sequences which are merged two by two again.
The process continues until there are only two optimal sequences left to merge, each with a size between 2 and $N/2$. The final merging produces an optimal sequence for computing **p-TV**u.
The algorithm below, named "Merge" performs the splitting into multiple couple sub-sequences. Subsequently, we combine them as previously described until there is just one sub-sequence left for computing **p-TV**u.

Algorithm 2b: Merge

Merge(u,p)

com Split the sequence u into two-point sequences.

com L is the sequence of all these sequences.

1. N:=size(u); $K := \text{Log}_2(N) - 1$

2. L:=empty list
3. **For** i= 1 to 2^K **do**
4. L[i]:=(u(2i-1),u(2i))
5. **End For**

com conquer sub-sequences two by two

6. **For** k= 1 to K **do**
7. M:= empty list
8. **For** i= 1 to 2^{K-k} **do**
9. M[i]:=Conquer(L[2i-1],L[2i])
10. **End For**
11. L:=M
12. **End For**
13. Result; L

Algorithm "MEI" The algorithm "MEI" is the algorithm "Merge" initialised by the algorithm "Eraser": "Merge by Eraser initialisation algorithm". "M" means "Merge", "E" "Eraser" and "I" "initialization". It was first proposed in [6]. The main idea is to reduce the cost of "Eraser" which try to merge not optimal sequences after each erasing and to improve the initialisation of "Merge".

The principle is the following: take a sequence $u = (u(0), ..., u(N))$ and start with the Eraser algorithm. Use Eraser algorithm to do all the tests $H_{ij}^{<}$ but after an erasing do not restart at the beginning. In this case two sub-lists are obtained and the tests can be applied independently on the two sub-lists. So, continue to use "Eraser" in the same way on the two sub-lists to obtain more and more lists. Finally the initial list in cut in many independent lists. Moreover, the sub-lists obtained are necessary optimal sequences since they satisfy the tests $H_{ij}^{<}$. Thus, it is an interesting departure to start to merge all these optimal sub-sequences. Indeed, we have many sub-lists optimized and less points that initially. We can then use the Merge algorithm to merge the list.

3.3 "AOP"=Add One Point

The last algorithm, "AOP", is the simplest to implement and seems to be faster than "Eraser" and "Merge" [11]. Numerical comparisons on thousands examples has be done in [24].
The basic principle is to merge an optimal sequence with a sequence with only one value and to extract of the new enlarged sequence an optimal subsequence. Let L be a sequence with the length l, to concanete this sequence with a ne w value v, $\widetilde{L} = (L(1), \ldots, L(l), v)$ and to extract from $\widetilde{L}$ a new optimal sequence $\overline{L}$, so **p-TV**$\widetilde{L} = p$-$\mathfrak{sum}$ $\overline{L}$.
This can been seen as a special merging. The big advantage of this merging is the cheap cost, very less than merging two optimal sequences with lengths $l/2$. Indeed, since L is an optimal

sequence, the only inequalities $H_{ij}^{<}(\widetilde{L})$ to check are the inequalities involving the last term v, i.e. $i+j=l+1$, so only $\mathscr{O}(l)$ inequalities.
Now, to extract an optimal subsequence to a given sequence u with length N, we start with the optimal sequence $L^1=(u(1))$ with length $l_1=1$, we concanete $u(2)$, $\widetilde{L}^2=(u(1),u(2))$ and extract for the new sequence an optimal one L^2.
At the step n, L^n in an optimal subsequence of $(u(1),\ldots,u(n)$, $1\leq n\leq N$. The length of L^n is $l_n\leq n$, $\widetilde{L}^{n+1}=(L^n(1),\ldots,L^n(l_n),u(n+1)$ is a subsequence of $(u(1),\ldots,u(n+1)$. We extract from $\widetilde{L}^{n+1}$ an optimal sequence L^{n+1}. At the end L^N is an optimal sequence of the whole sequence u.
Notice also, that the merging is simpler than in "Merge" or in 'Eraser". When we erase (extract a subsequence if $H_{ij}^{<}$ is false), we simply restart the process of checking all inequalities $H_{ij}^{<}$ involving the last term $u(n+1)$.
The algorithm can be shortened starting with a sequence with no constant step. Thus $(u(1),u(2))$ is an optimal sequence and we can start with $n=2$.
We also compute the p-sum more efficiently to reduce the cost of the algorithm.
Now, the algorithm can be written.

Algorithm 3: Add one point (AOP)

AOP(u,p)

1. N:=size(u)
2. Start with the list $L=(u_1,u_2)$.
3. **For** $n=3$ **to** N **do**
4. add the point u_n to the list L
5. $l=length(L)$; $S=0$
6. **for** $j=2$ **to** $l-1$, $j=2$ or j odd, **do**
7. $S:=S+|u(l+1-j)-u(l+2-j)|^p$
8. **if** $H_{l+1-j,j}^{\geq}$ **then**
9. $L:=(L(1),\ldots,L(l+1-j),u(n))$
10. Gotoline 5.
11. **end then**
12. **end do**
13. **End do**
14. Result: L

Notice that S is used to test inequality $H_{l+1-j,j}^{\geq}$. This test has usually a cost $\mathscr{O}(j)$ but with this computation of the sum, it reduces to a cost $\mathscr{O}(1)$.

4 Efficiency of the algorithms

In this section, the efficiency of the presented algorithms is discussed. All these algorithms has a polynomial costs. The polynomial cost of algorithms is now addressed more precisely.

4.1 The cost is at least N^2

Optimal criteria given by Proposition 2.5 has to be used to check that a sequence is an optimal one to compute its **p-TV** by its p-$\mathfrak{sum}$, i.e. **p-TV**$u = p$-$\mathfrak{sum}$ u.
Notice that, even if u is an optimal sequence there are many tests to do, indeed $\mathscr{O}(N^2)$ tests. Moreover the tests involve large sums with at most N terms. So, naively, the cost is at least $\mathscr{O}(N^3)$. It is possible to reduce the cost of sums because the sum involved in the inequality $H^<_{ij}$, $H^<_{(i+1)j}$ have the same terms except the first and the last one:

$$S_{ij} = \sum_{l=1}^{j} |u(i+l) - u(i+l-1)|^p,$$
$$S_{(i+1)j} = S_{ij} - |u(i+1) - u(i)|^p + |u(i+1+j) - u(i+j)|^p.$$

In this way, testing consecutive inequality $H^<_{ij}$ for increasing i does not need the computation of j additions with $j < N$ possibly large, but, only one substraction and one addition [24]. Managing well the computations of sums as in [24] for "AOP" and "Eraser", we can reduce the cost to the number of tests. Thus, $\mathscr{O}(N^2)$ is the minimal possible cost for any algorithms computing **p-TV**.

Preprocessing

Since the cost is a power of N, the number of the terms of the sequence, i.e. the length of the sequence, it is useful to reduce the size of the sequence before using one of the algorithms presented. These preprocessings are independent on p, so do not need computations of powers p and large sums.

1. u has no constant step. If u has some constant step, $u(i) = u(i+1)$ for some i, the sequence is reduced. The following preprocessing has a cost $\mathscr{O}(N)$.

   While $i <$ length of u do

   if $u(i) = u(i+1)$ then erase the value $u(i)$

   i:= i+1

   End

2. No monotonicity zone. If $(u(i-1) - u(i))(u(i) - u(i+1)) \geq 0$ then, the midle term $u(i)$ is removed without changing **p-TV**u. This process corresponds to satisfy the condition $H_{\cdot 3}$ without computing the sum and the three p-powers to check inequality $H^<_{i2}$. It means that if three consecutive terms are in monotinc order, the midle term is removed. Again, The following preprocessing has a cost $\mathscr{O}(N)$.

Optimal sequence, most costly?

The cost of the algorithm depends on the algorithm, the size of the sequence, but also on the sequence itself. The cost is called $C(A, N, u)$, A stands for the algorithm and u for the sequence with N terms. For instance, if the sequence is constant or increasing, the cost is only N. This is the order of the price of the preprocessing.

We consider here the maximal cost $C(A, N) = \max_u C(A, N, u)$.

Many numerical test suggest that the cost is maximal when the sequence is already optimal[24]. It seems that is a consequence of the supposed power-law cost with a power at least 2.

There is an heuristic explanation of this fact. To explain this fact, one algorithm is chosen, "Eraser" for instance. Let $C(N, u) = C("Eraser", N, u)$. Forgetting the multiplicative constant giving $C(N, u)$, assume that $C(N, u) = N^2$ when the initial sequence is already optimal, i.e. no erasing. No compare this cost when there is only one erasing, i.e. one subsequence extracted to obtain an optimal sequence. Due to the preprocessing, if some points are erased to the initial sequence there are at least two points erased. The initial sequence has no constant step and only local extrema. Consider the case with 2 points erased, then the subsequence v has $N - 2$ terms. That means that many inequalities $H^{\leq}_{i3}$ have been done before the erasing of two points. Loosely speaking, let's say this cost is N because there are less than N tests to do. Finally, the cost to check that v is an optimal sequence is the cost to find which two consecutive points in u have to be erased and the cost to check that v with $N - 2$ terms is an optimal sequence. That is,

$$C(N, v) = N + (N - 2)^2 = N^2 - 3N + 4 < C(N) = N^2.$$

The inequality is at least true for N large enough.

We suspect that the optimal sequence are the most costly or at least, such sequences give good insight about the usual complexity of the algorithms.

As a consequence, the complexity of the algorithms are next studied only for an optimal sequences. Further studies are needed to obtain the cost of the algorithms for the general case.

4.2 Cost comparisons of the three algorithms

In this section, the complexity of the three algorithms are done on the most expensive suspected sequence, that is to say when the sequence is already optimal. The cost is counted with the number of tests done. We suppose that the computations of p-$\mathfrak{sum}$ are optimised as it is discussed above and as it was done in [24] for "AOP" and "Eraser". The optimisation of p-$\mathfrak{sum}$ is more complicated for "Merge". Next, a comparison of these algorithms is discussed.

Eraser

The cost of Eraser on an optimal sequence is simply the cost to check all inequalities in Proposition 2.5. Thus the cost is N^2 in this case.

AOP

We compute the number of tests that "AOP" does on an optimal sequence. All the tests with k points means to check inequality $H^{\leq}_{\cdot j}$ with $j = k - 1$.

$$\begin{array}{cc} \text{test 3 pts} & 2, \\ \text{test 4 pts} & 3, \\ \text{test 6 pts} & 5, \\ \vdots & \\ \text{test k pts} & k-1, \\ \vdots & \\ \text{test N pts} & N-1. \end{array}$$

Therefore,

$$2+3+5+7+9+...+(N-1)) = 2+\sum_{j=2}^{N/2}(2j-1) \sim \sum_{j=2}^{N/2}(2j-1) = \frac{N/2-1}{2}(N+2) \sim \frac{N^2}{4}.$$

Like Eraser, the cost is of order N^2.

Merge

To compute the cost of "Merge", we first have to compute the cost of the merging of two optimal sequences with the same size M yielding to an optimal sequence. We consider only this case since the cost of "Merge" is done on an optimal sequence u, and any subsequence v of consecutive terms of u is also optimal. Then we have $3+5+...+M-1$ tests to do which yield to a total number of tests of order M^2.
Now, the cost of "Merge" for an optimal sequence with $N=2^k$ terms is the number of operations for a merging times the number of merging.

$$\begin{array}{cc} \text{The first merging} & \left(2^1\right)^2 \cdot (2)^{k-2}, \\ \text{The second merging} & \left(2^2\right)^2 \cdot (2)^{k-3}, \\ \vdots & \vdots \\ \text{The last merging} & \left(2^{k-1}\right)^2 \cdot (2)^0. \end{array}$$

Therefore,

$$\sum_{j=1}^{k-1}\left(2^j\right)^2 \cdot 2^{k-j-1} = 2^{k-1}\sum_{j=1}^{k-1}2^j = 2^{k-1}\cdot 2\cdot\frac{1-2^{k-1}}{1-2} = N\cdot\left(\frac{N}{2}-1\right) \sim \frac{N^2}{2}.$$

Again, "Merge" has a cost of order N^2.

Notice that in "Merge" the last merging with two sequences with $N/2$ points has a cost with the same order has the whole "Merge" algorithm. The final merging is then the principal part of the "Merge" cost.

Crude bounds for the cost

For an optimal sequence, the cost N^2 is founded when the p-$\mathfrak{sum}$ are efficiently computed, else the complexity is of order N^3.
What happens if the initial sequence is not optimal? That is when there are some erasing. Let us give a crude bounds to show that the complexity stays polynomial but with a possible increase

of the power. Assume that the computations of p-$\mathfrak{sum}$ are optimised. The same argument works for p-$\mathfrak{sum}$ not optimised with a greater power.
Let u be a sequence with N terms and suppose there are an erasing with k points and then no erasing. The cost to find the points to erase is less than the cost to check all the inequalities $H_{ij}^{<}$, that is N^2. Next the subsequence is supposed optimal, so there are $(N-k)^2$ tests to do. Finally the total cost is bounded by $N^2+(N-k)^2$.
If there are e erasing then, by a similar argument, the total cost is less than

$$\begin{aligned} &N^2+(N-k_1)^2+(N-k_1-k_2)^2+\ldots(N-k_1\ldots-k_e)^2\\ &\leq N^2+(N-1)^2+\ldots+(N-(N-1))^2\sim N^3/3. \end{aligned}$$

Hence the cost for ageneral subsequence is at most N^3. This is a crude upper bound and the authors expect that the real cost is less or the worst case is very rare.
Of course when the p-$\mathfrak{sum}$ are not optimised, the cost is of order N^4.

Comparisons of the 3 algorithms
Preliminary numerical tests are done in [24]. "AOP" seems the best algorithm, followed by "Eraser" and finally by "Merge". Notice that, "AOP" is easier to programm than "Eraser" , and "Eraser" is even easier to programm than "Merge", thus there are room to improve the first studies given in the report [24]. A possibility is the preliminary study in [24] can only reflect that optimising the code for "AOP" is easy, for "Eraser" less easier, and more difficult for "Merge". Anyway, this preliminary study of the cost needs further mathematical and numerical works.

A Motivations to compute the p-variation

History on p-variation The p-variation is first used by Norbert Wiener [33] for $p=2$ to get the convergence of Fourier series following Dirichlet for piecewise C^1 functions and then Jordan for BV function. Such spaces are generalized for all p in [25, 35] and then generalized variation with a convex function more general than a power law in [36]. For the theory of p-variation and generalized variation, we suggest to books [5, 13, 25] after the article [28] build a nice metric structure for generalized variation.

The p-variation is mainly used in integration theory and probabilities for a long time [17]. Some applications in probabilities can be found in [14, 29, 32] mainly supported by Rimas Norvaisa and his collaborators.

PDEs, hyperbolic conservation laws Here, the main motivations of the authors are related to a new subject where the p-variation spaces can be useful. The theory of hyperbolic partial differential equations (PDEs) with nonlinear flux is usually studied in the space BV. Recently the second author and more and more collaborators introduce the fractional BV spaces which are indeed the p-variation spaces [2].

It is well known that shock waves appear in finite time even when the initial data is smooth. It is the reason why the BV space of function with bounded total variation is often used. It allows us to consider weak discontinuous solutions with right and left limits everywhere. Moreover, the embedding from BV in L^1_{loc} is compact. This is fundamental to prove the global existence of weak solutions for such conservation laws. But all weak solutions do not belong to BV. It is the reason why fractional BV spaces, BV^s, with $0<s<1$ has introduced in [2] in the context of conservation laws. These spaces have the same properties as BV for the traces and

the compactness. These spaces are larger than BV space. The exponent "s" corresponds to the fraction Sobolev derivative and $BV^1 = BV$. Since 2014 there are more and more applications of these BV^s spaces for equations [7, 19, 18], and for systems of conservation laws [3, 4, 22], existence theory, regularity, and blow-up. All these results are theoretical and the BV^s norm is not computed.
Now, there is a numerical motivation to look for numerical schemes preserving the fractional BV estimates so to compute the p-variation. Today, only the Godunov scheme [1, 3, 10] and some Wave Front Tracking Algorithms [4, 18, 21] are already proven to be compatible with the p-variation for scalar conservation laws and some systems.

References

[1] Maryam Al Zohbi, Stéphane Junca. "Entropy solutions to a non-conservative and non-strictly hyperbolic diagonal system inpired by dislocation dynamics". *Journal of Evolution Equations*, 2023, in press. hal-03849987

[2] Christian Bourdarias, Marguerite Gisclon and Stéphane Junca. "Fractional BV spaces and first applications to scalar conservation laws". *Journal of Hyperbolic Differential Equations*, 2014, 11 (4), pp.655-677.

[3] C. Bourdarias, M. Gisclon, S. Junca, and Y.-J. Peng. "Eulerian and Lagrangian formulations in BV^s for gas-solid chromatography". *Commun. Math. Sci.* 14 (2016), no. 6, 1665–1685.

[4] Christian Bourdarias, Anupam Pal Choudhury, Billel Guelmame and Stéphane Junca. "Entropy solutions in BV^s for a class of triangular systems involving a transport equation". *SIAM J. Math. Anal.* 54 (2022), no. 1, 791–817.

[5] Michel Bruneau. *Variation totale d'une fonction.* (French) Lecture Notes in Mathematics, Vol. 413. Springer-Verlag, Berlin-New York, 1974. xiv+332 pp.

[6] Vygantas Butkus and Rimas Norvaiša. "Computation of p-variation". *Lith. Math. J.* 58 (2018), no. 4, 360–378.

[7] Pierre Castelli and Stéphane Junca. "Oscillating waves and optimal smoothing effect for one-dimensional nonlinear scalar conservation laws". Hyperbolic problems: theory, numerics, applications, 709–716, AIMS Ser. Appl. Math., 8, Am. Inst. Math. Sci. (AIMS), Springfield, MO, 2014.

[8] Pierre Castelli, Stéphane Junca. "Smoothing effect in BV_Φ for entropy solutions of scalar conservation laws". *J. Math. Anal. Appl.* 451 (2017), no. 2, 712–735.

[9] Pierre Castelli, Stéphane Junca. "On the maximal smoothing effect for multidimensional scalar conservation laws". *Nonlinear Analysis: Theory, Methods and Applications*, 155 (2017), pp.207-218

[10] Anupam Pal Choudhury, Stéphane Junca. "Decay of p-variation for Godunov scheme". preprint 2023.

[11] Aimen Daoudi. Algorithms for fractional BV norms. Master Thesis, June 2022, Université Côte d'Azur, 47 pp.

[12] Giuseppe De Marco. "Representing the algebra of regulated functions as an algebra of continuous functions". *Rend. Sem. Mat. Univ. Padova* 84 (1990), 195–199 (1991).

[13] Richard Mansfield Dudley and Rimas Norvaiša. *Differentiability of Six Operators on Nonsmooth Functions and p-Variation*. Lecture Notes in Mathematics, 1703. Springer-Verlag, Berlin, 1999. viii+277 pp.

[14] Richard Mansfield Dudley and Rimas Norvaiša. *Concrete functional calculus*. Springer Monographs in Mathematics. Springer, New York, 2011. xii+671 pp.

[15] Casper Goffman, Gadi Moran and Daniel Waterman. "The structure of regulated functions". *Proc. Amer. Math. Soc.* 57(1) (1976) 61–65.

[16] Dana Fraňková. "Regulated functions". *Math. Bohem.* 116 (1991), no. 1, 20–59.

[17] Peter K. Friz, Nicolas B. Victoir. Multidimensional stochastic processes as rough paths. Theory and applications. Cambridge Studies in Advanced Mathematics, 120. Cambridge University Press, Cambridge, 2010. xiv+656 pp.

[18] Billel Guelmame, Stéphane Junca and Didier Clamond. "Regularizing effect for conservation laws with a Lipschitz convex flux". *Commun. Math. Sci.* 17 (2019), no. 8, 2223–2238.

[19] Shyam Sundar Ghoshal, Billel Guelmame, Animesh Jana and Stéphane Junca. "Optimal regularity for all time for entropy solutions of conservation laws in BV^s". *NoDEA Nonlinear Differential Equations Appl.* 27 (2020), no. 5, Paper No. 46, 30 pp.

[20] Boris Haspot, Stéphane Junca. Fractional BV solutions for 2×2 system of conservation laws with a genuinely nonlinear field and a linearly degenerate field. preprint 2022, hal-02532444

[21] Helge Kristian Jenssen and Johanna Ridder. "On ϕ-variation for 1-d scalar conservation laws". *Journal of Hyperbolic Differential Equations* Vol. 17, No. 4 (2020) 843–861.

[22] Stéphane Junca and Bruno Lombard. "Analysis of a Sugimoto model of nonlinear acoustics in an array of Helmholtz resonators". *SIAM J. Appl. Math.* 80 (2020), no. 4, 1704–1722.

[23] Ronald J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 2002. xx+558 pp.

[24] Hanwen Li, Tom Elbaz, Stéphane Junca. " Efficiency of some algortihms to compute fractal total variation". Research report, Université Côte d'Azur, (2023), 18 pp. hal-04025611

[25] E.-R. Love, Laurence Chisholm Young. "Sur une classe de fonctionnelles linéaires". *Fund. Math.* 28 (1937), 243-257.

[26] Elio Marconi. "Regularity estimates for scalar conservation laws in one space dimension". *J. Hyperbolic Differ. Equ.* 15 (2018), no. 4, 623–691.

[27] Elio Marconi. "Structure and regularity of solutions to 1D scalar conservation laws". Hyperbolic problems: theory, numerics, applications, 546–553, AIMS Ser. Appl. Math., 10, Am. Inst. Math. Sci. (AIMS), Springfield, MO, [2020].

[28] Julian Musielak and Wladyslaw Orlicz. "On generalized variations". *I. Studia Math.* 18 (1959), 11-41.

[29] Rimas Norvaiša. "Rough functions: p-variation, calculus and index estimation". *Lithuanian Math. J.* 46 (2006), no. 1, 102–128

[30] Rimas Norvaiša and Alfredas Rackauskas. "Convergence in law of partial sum processes in p-variation norm". *Lithuanian Math. J.* 48 (2008), no. 2, 212–227.

[31] Rimas Norvaiša and Dona Mary Salopek. "Estimating the p-variation index of a sample function: An application to financial data set". *Methodol. Comput. Appl. Probab.* 4 (2002), no. 1, 27–53.

[32] Jinghua Qian. "The p-variation of partial sum processes and the empirical process". *Ann. Probab.* 26 (1998), no. 3, 1370–1383.

[33] Norbert Wiener. " The quadratic variation of a function and its Fourier coefficients". *J. Math. Phys. Mass. Inst. Techn.* 3 (1924), no. 2, 72–94.

[34] "Divide and conquer Algorithm". Wikipedia, last consultation 29-11-2023.

[35] Laurence Chisholm Young. "An inequality of the Hölder type, connected with Stieltjes integration". Acta Math. 67 (1936), no. 1, 251–282.

[36] Laurence Chisholm Young. "General inequalities for Stieltjes integrals and the convergence of Fourier series". *Math. Ann.* 115 (1938), no. 1, 581–612.

January 14, 2024