ONLINE MULTIVARIATE CHANGEPOINT DETECTION: LEVERAGING LINKS WITH COMPUTATIONAL GEOMETRY

Liudmila Pishchagina
Université Paris-Saclay, CNRS, Univ Evry
Laboratoire de Mathématiques et Modélisation d'Evry
91037, Evry-Courcouronnes, France
liudmila.pishchagina@univ-evry.fr

Gaetano Romano
Department of Mathematics and Statistics,
Lancaster University, Lancaster LA1 4YF, UK
g.romano@lancaster.ac.uk

Paul Fearnhead
Department of Mathematics and Statistics,
Lancaster University, Lancaster LA1 4YF, UK
p.fearnhead@lancaster.ac.uk

Vincent Runge
Université Paris-Saclay, CNRS, Univ Evry
Laboratoire de Mathématiques et Modélisation d'Evry
91037, Evry-Courcouronnes, France
vincent.runge@univ-evry.fr

Guillem Rigaill
Université Paris-Saclay, CNRS, Univ Evry
Laboratoire de Mathématiques et Modélisation d'Evry
91037, Evry-Courcouronnes, France

Université Paris-Saclay, CNRS, INRAE, Univ Evry
Institute of Plant Sciences Paris-Saclay (IPS2)
Orsay, France
guillem.rigaill@inrae.fr

November 3, 2023

ABSTRACT

The increasing volume of data streams poses significant computational challenges for detecting changepoints online. Likelihood-based methods are effective, but their straightforward implementation becomes impractical online. We develop two online algorithms that exactly calculate the likelihood ratio test for a single changepoint in p-dimensional data streams by leveraging fascinating connections with computational geometry. Our first algorithm is straightforward and empirically quasi-linear. The second is more complex but provably quasi-linear: $\mathcal{O}(n \log(n)^{p+1})$ for n data points. Through simulations, we illustrate, that they are fast and allow us to process millions of points within a matter of minutes up to $p = 5$.

Keywords computational geometry, convex hull, dynamic programming, functional pruning, online changepoint detection

1 Introduction

In recent years, many methods have been proposed for detecting one or multiple changepoints offline or online in data streams. The reason for such a keen interest in changepoint detection methods lies in its importance for various real-world applications, including bioinformatics [1, 2], econometrics [3, 4], medicine [5, 6, 7], climate and oceanography [8, 9, 10, 11, 12], finance [13, 14], autonomous driving [15], entertainment [16, 17, 18], computer vision [19] or neuroscience [20]. Whether this is online (i.e. observations are processed as they become available) or offline (i.e.

observations are processed only once the whole data is available) the ever-increasing size of the data stream raises computational challenges. Ideally, a changepoint procedure should be linear or quasi-linear in the size of the data and, for online applications, should also be able to process new observations as quickly as they arrive. A common idea for both offline and online parametric (multiple) changepoint model estimation, is to optimise a likelihood function or calculate likelihood ratio statistics [21, 22]. From a computational perspective this optimisation can either be done (1) heuristically using, for example, Page CUSUM [23] in an online context or Binary Segmentation and its variants [24, 14] in an offline context, or (2) exactly using dynamic programming techniques as in the Segment Neighbourhood, or Optimal Partitioning algorithms [22, 25]. Importantly, these later algorithms used with an appropriate penalty enjoy good statistical properties [26, 27, 28, 29] and have shown good properties in numerous applications [30, 31].

Pruning ideas have emerged in the literature to speed these dynamic programming algorithms without sacrificing their exactness [10, 32, 33, 34]. Two types of rules have been proposed. Inequality pruning rules, as in the PELT algorithm [10], apply to a wide range of changepoint models but are efficient only if the number of true changepoints increases quickly with the size of the data. Functional pruning, as in the pDPA, FPOP or FOCuS [32, 33, 34] algorithms, applies mostly to univariate models but is efficient even when the number of true changepoints is small in comparison to the size of the data. Interestingly, these algorithms are sequential in nature and empirically quasi-linear. This quasi-linearity was recently proven for the online detection of a single changepoint [34]. The pDPA, FPOP, and FOCuS algorithms rely on functionalisation of the problem, that is they consider the likelihood as a function of the last segment parameter. A direct extension of these ideas to a higher dimension ($p > 1$) has proven difficult because it involves computing the intersection and subtraction of many convex sets in $\mathbb{R}^p$ [35, 36].

In this paper, we study parametric online models with one changepoint, where the distributions are drawn from the p-variate exponential family. We extend the functional problem using a linearisation argument. This extension allows us to demonstrate a link between the functional problem and problems studied in computational geometry. In particular, we show that the solutions of this extension correspond to identifying the supporting hyperplanes of a particular half-space intersection problem and demonstrate that this problem further simplifies and is dual to a convex hull problem.

The paper has the following structure. Section 2 describes the maximum likelihood framework we are using. Section 3 presents the mathematical formulation of the functional pruning problem and how it is related to a half-space intersection problem and a convex hull problem. In Section 4, the theory from Section 3 is applied to the online detection of a single changepoint with known or unknown pre-changepoint parameter. The proposed method, called MdFOCuS, is introduced in Subsection 4.3. In Section 4.3.2, empirical efficiency comparisons of MdFOCuS in dimension p with $p\times$FOCuS [34] and ocd [37] are performed on simulated data. In Section 5 we apply our methodology to track the performances of an NBA team and discuss how our theory could accelerate the recently proposed non-parametric e-detectors [38].

2 Maximum Likelihood Methods for Online Single Changepoint Detection

2.1 Problem set-up

We consider observations $\{y_t\}_{t=1,2,...}$ in $\mathbb{R}^{p'}$ and wish to detect the possible presence of a changepoint and estimate its location. If there is a change, we denote the time of the change as τ. We will consider likelihood-based methods for doing this, which involve postulating a model for the data within each segment between consecutive data points. Our model will be that the data point y_t is independent and identically distributed from a distribution with density $f_Y(y_t|\theta_1)$ before the change and $f_Y(y_t|\theta_2)$ after the change.

At time n twice the log-likelihood for the time series $\{y_t\}_{t=1,...,n}$, in terms of the pre-changepoint parameter, θ_1 in $\mathbb{R}^{p''}$, the post-changepoint parameter, θ_2 in $\mathbb{R}^{p''}$, and the location of a change, τ, is

$$\ell_{\tau,n}(\theta_1,\theta_2) = 2\left(\sum_{t=1}^{\tau}\log f_Y(y_t|\theta_1) + \sum_{t=\tau+1}^{n}\log f_Y(y_t|\theta_2)\right). \qquad (1)$$

Depending on the application one may assume that the pre-changepoint parameter θ_1 is either known or unknown. In either case, we can get the log-likelihood for a changepoint at τ by maximising this equation over the unknown parameter, so θ_2 if the pre-changepoint parameter is known or both θ_1 and θ_2 otherwise. In practice we do not know the changepoint location, so we would maximize this log-likelihood over τ. We can define the maximum likelihood estimator for τ, for the pre-changepoint parameter being either known or unknown respectively, as

$$\hat{\tau}(\theta_1) = \arg\max_{\tau}(\max_{\theta_2}\ell_{\tau,n}(\theta_1,\theta_2))\,,$$
$$\hat{\tau}(.) = \arg\max_{\tau}(\max_{\theta_1,\theta_2}\ell_{\tau,n}(\theta_1,\theta_2)).$$

We then obtain a Generalized Likelihood Ratio (GLR) statistic [21] for a change τ by comparing with twice the log-likelihood for a model with no change. That is $2\sum_{t=1}^{n} f_Y(y_t|\theta_1)$ if θ_1 is known and $\max_{\theta_1} 2\sum_{t=1}^{n} f_Y(y_t|\theta_1)$ if it is not.

Offline, optimising equation (1) is rather simple. Indeed assuming we have access to all the data it suffices to compute, for every possible changepoint location from 1 to $n-1$, the likelihood (which can be done efficiently using summary statistics) and then pick the best value. Online, the problem is substantially more difficult as one needs to recompute the maximum for every new observation. Applying the previous offline strategy for every new observation is inefficient as it takes order n to process the nth observation. It is infeasible for an online application, where n grows unbounded, and for this reason various heuristic strategies have been proposed (see [39] for a recent review). From a computational perspective, we roughly identify two main types of heuristics. The first idea is to restrict the set of possible (θ_1, θ_2) pairs to be discrete. The simplest case is to consider just one pair of (θ_1, θ_2) as in the Page CUSUM strategy [23] but one typically considers a grid of values [37]. The second idea is to restrict the set of changepoint locations one considers. The simplest case is arguably to consider a grid of window lengths $w_1, \ldots, w_M$ and at time n compute the GLR statistic only for changes at $\tau = n - w_1, \ldots, n - w_M$ [40, 41]. But this naturally introduces a trade-off between computational efficiency and statistical power: the larger the grid of windows, the larger the computational overhead [39].

Recently, the FOCuS algorithm has been introduced that calculates the maximum likelihood and likelihood ratio test exactly but with a per-iteration cost that increases only logarithmically with the amount data [34, 42] - however this algorithm is only applicable for detecting changes in a univariate feature of a data stream. A key insight from the FOCuS algorithm is that to maximise the likelihood one only needs to consider changes that correspond to points on a particular convex hull. This link to geometrical features of the data is one that we leverage later in developing algorithms for detecting changes in multivariate features.

2.2 Detection of a single change in the parameter of distribution in the exponential family

We briefly introduce our basic notations. We use $\mathbf{1}_p$ to denote a vector of length p in which all elements are set to one. For any vector $\mu = (\mu^1, \ldots, \mu^p)$ in $\mathbb{R}^p$, we define its q-norm as $||\mu||^q := \left(\sum_{k=1}^{p} |\mu^k|^q\right)^{1/q}$, where $(q > 1)$ in $\mathbb{R}$. For two vectors, $\mu_1 = (\mu_1^1, \ldots, \mu_1^p)$ and $\mu_2 = (\mu_2^1, \ldots, \mu_2^p)$ in $\mathbb{R}^p$, we define their scalar product as $\langle \mu_1, \mu_2 \rangle = \sum_{k=1}^{p} \mu_1^k \mu_2^k$. The symbol $|\cdot|$ signifies the cardinality of $\cdot$ depending on the context.

We are now ready to describe our model. We assume that the density $f_Y(y_t|\theta_k)$, for observation y_t in segment $k \in \{1, 2\}$ which has segment-specific parameter θ_k, can be written in the form

$$f_Y(y_t|\theta_k) = \exp\left\{-\frac{1}{2}\left[A'(r(\theta_k)) - 2\langle r(\theta_k), s(y_t)\rangle + B'(s(y_t))\right]\right\}, \tag{2}$$

where s and r are functions mapping respectively the observations, y_t, and the parameters, θ_k, to $\mathbb{R}^p$, A' is some convex function from $\mathbb{R}^p$ to $\mathbb{R}$, and B' from $\mathbb{R}^p$ to $\mathbb{R}$. Defining the natural parameter η_k as $\eta_k = r(\theta_k)$, and x_t as $x_t = s(y_t)$ we rewrite this density as a function of η_k and x_t:

$$f_X(x_t|\eta_k) = \exp\left\{-\frac{1}{2}\left[A'(\eta_k) - 2\langle \eta_k, x_t\rangle + B'(x_t)\right]\right\}. \tag{3}$$

This class encompasses many common exponential family models, see Table 1 for examples of several models of changes in the mean of p-variate data (where $p = p' = p''$) and the corresponding functions s, r, A' and B'. Later we will model a change in the mean (θ_1) and variance (θ_2) (so $p'' = 2$) of a Gaussian signal in $\mathbb{R}$ (so $p' = 1$). In this slightly more complex case we have $p' \neq p = 2$ and we can define s, r, A' and B' as follows

$$\theta = (\theta_1, \theta_2), \quad \eta = \left(\frac{\theta_1}{\theta_2}, -\frac{1}{2\theta_2}\right), \quad s(y) = (y, y^2), \quad A'(\eta) = \frac{\theta_1^2}{\theta_2} - \log(\theta_2), \quad B'(x) = \log(2\pi).$$

Let us denote the pre-changepoint natural parameter as η_1 and the post-changepoint natural parameter by η_2. We can then define the log-likelihood of our (transformed) data $x_1, \ldots, x_t$, under a model with a change at τ as

$$\ell_{\tau,n}(\eta_1, \eta_2) = -\sum_{t=1}^{\tau}[A'(\eta_1) - 2\langle x_t, \eta_1\rangle + B'(x_t)] - \sum_{t=\tau+1}^{n}[A'(\eta_2) - 2\langle x_t, \eta_2\rangle + B'(x_t)]. \tag{4}$$

The log-likelihood estimator for τ, for the pre-change parameter being either known or unknown respectively, is

$$\hat{\tau}(\eta_1) = \arg\max_{\tau}(\max_{\eta_2} \ell_{\tau,n}(\eta_1, \eta_2)),$$
$$\hat{\tau}(.) = \arg\max_{\tau}(\max_{\eta_1,\eta_2} \ell_{\tau,n}(\eta_1, \eta_2)).$$

Table 1: Modelling a change in the mean of a p-variate data with independent Gaussian (known variance), Poisson, Binomial and Exponential errors. Examples of distributions from the natural exponential family with a parameter θ in $\mathbb{R}^p$ and the corresponding forms of the functions s, r, A' and B'. Without loss of generality the variance of the Gaussian model is assumed to be 1 for all dimensions. The number of trials is assumed to be m for all dimensions in the Binomial model.

Distribution	$\eta := r(\theta)$	$x := s(y)$	$A'(\eta)$	$B'(x)$
Gaussian (changes in mean)	θ	y	$\lVert\eta\rVert^2$	$\lVert x\rVert^2 + \log(2\pi)^p$
Poisson	$\log\theta$	y	$2\langle e^\eta, \mathbf{1}_p\rangle$	$2\langle \log x!, \mathbf{1}_p\rangle$
Binomial	$\log\frac{\theta}{1-\theta}$	y	$2m\langle \log(1+e^\eta), \mathbf{1}_p\rangle$	$-2\left\langle \log\binom{m}{x}, \mathbf{1}_p\right\rangle$
Exponential	$-\theta$	y	$-2\langle\log(-\eta), \mathbf{1}_p\rangle$	0

The GLR statistic for a change τ is obtained by comparing with $\sum_{t=1}^n [A'(\eta_1) - 2\langle x_t, \eta_1\rangle + B'(x_t)]$ if η_1 is known and with $\max_{\eta_1} \sum_{t=1}^n [A'(\eta_1) - 2\langle x_t, \eta_1\rangle + B'(x_t)]$ if it is not.

For any fixed η_1 and any η_2 the addition of $\sum_{t=1}^n [A'(\eta_2) - 2\langle x_t, \eta_2\rangle + B'(x_t)]$ to (4) does not change the arg-maximiser: $\arg\max_\tau \ell_{\tau,n}(\eta_1, \eta_2)$. Moreover, this addition eliminates the dependence on n. Hence when searching for the maximum likelihood change we can redefine $\ell_{\tau,n}(\eta_1, \eta_2)$ as follows:

$$\ell_\tau(\eta_1, \eta_2) = \tau\,[A'(\eta_2) - A'(\eta_1)] - 2\left\langle \sum_{t=1}^{\tau} x_t, \eta_2 - \eta_1 \right\rangle, \tag{5}$$

and be sure that the optimal change optimises $\ell_\tau(\eta_1, \eta_2)$ at least for a pair (η_1, η_2). For all subsequent proofs, it will suffice to study the special case where η_1 is known. Our results for the unknown case (on $\hat{\tau}(.)$ in Theorem 5) are derived as a by-product: considering all possible η_1 simultaneously. Considering η_1 to be fixed we thus introduce the following definitions:

$$\mu = \eta_2 - \eta_1, \quad A(\mu) = A'(\mu + \eta_1) - A'(\eta_1),$$

and rewrite the function $\ell_\tau(\eta_1, \eta_2)$ as a function of μ:

$$\ell_\tau(\mu) = \tau A(\mu) - 2\left\langle \sum_{t=1}^{\tau} x_t, \mu \right\rangle. \tag{6}$$

3 Linear Relaxation of the Functional Pruning Problem

3.1 Mathematical Formulation

Functional pruning ideas give the potential of an efficient approach to recursively calculate the GLR statistic for a change at time $n+1$ given calculations at time n [34]. The idea is to restrict the set of possible changepoints prior to $n+1$ that we need consider. Working out which changepoint locations we need to consider is obtained by viewing the GLR statistic as a function of the post-changepoint parameter μ, and finding the set of changepoint locations which contribute to this, i.e. for which the GLR statistic is maximum for some value of μ.

In more details, at step n of a functional pruning algorithm, we consider a list of possible changepoints τ of the data from x_1, $x_2 \ldots$ to x_n. Each changepoint is represented by a function f_τ of the last segment parameter μ and a set of μ values for which it is maximising the likelihood: $Set_\tau = \{\mu | \forall \tau' \neq \tau,\ f_\tau(\mu) > f_{\tau'}(\mu)\}$. To proceed to the next data point $n+1$ the update is informally done as follows.

1. We include in the list of segmentations the one with a last-change at n which is represented by a constant function f_n.
2. We compare all other functions to this new function and accordingly restrict the set of μ for which they are optimal. That is we compute the intersection $Set_\tau \leftarrow Set_\tau \cap \{\mu | f_\tau(\mu) > f_n(\mu)\}$ for all $\tau < n$, and the intersection $Set_n \leftarrow \cap_\tau \{\mu | f_n(\mu) > f_\tau(\mu)\}$.
3. We discard all changepoints whose set Set_τ is empty.
4. We add the likelihood of the last data point $n+1$ to all functions f_τ.

The left column of Figure 1 provides a graphical representation of these steps. The pruning of the third step is valid because whatever the data points after $n+1$ might be these functions (or equivalently changepoints) can never maximize the log-likelihood. Formally, we now consider a set $\mathcal{T}$ indexing functions f_τ such that

$$f_\tau(\mu) = a_\tau A(\mu) - 2\langle \mu, b_\tau \rangle + c_\tau \,, \tag{7}$$

where a_τ, c_τ are in $\mathbb{R}$, and b_τ is in $\mathbb{R}^p$ and A is a convex function from $\mathcal{D}_A$, a subset of $\mathbb{R}^p$. We define $Im(A)$ as the image of function A. The idea is that at the current step of a functional pruning algorithm we are storing functions of the last segment parameter, here denoted by μ. Function f_τ corresponds to the log-likelihood for a segmentation with a change at τ and with parameter μ. This function will depend on μ through the log-likelihood contribution of the data in this segment, which will impact all the coefficients and the maximum log-likelihood for data in earlier segments. The functional form of f_τ will depend on the specific likelihood model. As an example, for detecting change in the mean of Gaussian data with $\eta_1 = 0$, we would have $A(\mu) = ||\mu||^2$, with $Im(A) = \mathbb{R}^+$ and $\mathcal{D}_A = \mathbb{R}^p$.

We then define

$$F_{\mathcal{T}}(\mu) = \max_{\tau \in \mathcal{T}} \{f_\tau(\mu)\} \,,$$

and, as informally stated, our goal is to recover all the indices τ maximising $F_{\mathcal{T}}$ for at least one value of μ. Formally, we define:

$$\mathcal{F}_{\mathcal{T}} = \{\tau | \exists \mu \in \mathcal{D}_A \subset \mathbb{R}^p, \forall \tau' \in \mathcal{T} \text{ with } \tau' \neq \tau : \; f_\tau(\mu) > f_{\tau'}(\mu)\}. \tag{8}$$

Remark 1. *Index set $\mathcal{F}_{\mathcal{T}}$ is defined using a strict inequality. As function A is strictly convex, (and assuming that for all $\tau \neq \tau'$ we cannot have simultaneously $a_\tau = a_{\tau'}$, $b_\tau = b_{\tau'}$ and $c_\tau = c_{\tau'}$) equality can be obtained only over a set of zero measure, consequently this case can be ignored.*

3.2 Linearisation of the Functional Pruning Problem

The size of the set $\mathcal{F}_{\mathcal{T}}$ depends on the nature of the function A. In essence, to consider any possible function A, we introduce a new parameter λ to replace $A(\mu)$. This way we get a problem that does not depend on the form of A and, as we will see, is easier to solve, albeit at the cost of an additional dimension. To be specific, for any function $\mu \mapsto f_\tau(\mu)$ we associate an (augmented) function $(\lambda, \mu) \mapsto g_\tau(\lambda, \mu)$ defined as:

$$g_\tau(\lambda, \mu) = a_\tau \lambda - 2\langle \mu, b_\tau \rangle + c_\tau.$$

As for the f_τ, we define the maximum $G_{\mathcal{T}}(\lambda, \mu)$ as

$$G_{\mathcal{T}}(\lambda, \mu) = \max_{\tau \in \mathcal{T}} \{g_\tau(\lambda, \mu)\}.$$

Again our goal will be to recover the index τ strictly maximising $G_{\mathcal{T}}$:

$$\mathcal{G}_{\mathcal{T}} = \{\tau | \exists (\lambda, \mu) \in Im(A) \times \mathcal{D}_A, \forall j \in \mathcal{T} \text{ with } j \neq i : \; g_\tau(\lambda, \mu) > g_{\tau'}(\lambda, \mu)\}.$$

In the previous definition, the parameter λ is constrained to be in the image of the function A. The values attained by f_τ are also attained by g_τ on a parametric curve of the (λ, μ) space given by equation $\lambda = A(\mu)$, so that $f_\tau(\mu) = g_\tau(A(\mu), \mu)$. This trivially leads to the following result that the set of τ strictly maximising $\mathcal{G}_{\mathcal{T}}$ must contain the set strictly maximising $\mathcal{F}_{\mathcal{T}}$.

Theorem 1. $\mathcal{F}_{\mathcal{T}} \subseteq \mathcal{G}_{\mathcal{T}}$.

Without any further assumptions on the function A, the set $\mathcal{G}_{\mathcal{T}}$ could be much larger than $\mathcal{F}_{\mathcal{T}}$. The following theorem demonstrates that equality holds for strictly convex power functions.

Theorem 2. *If for all $\tau \in \mathcal{T}$, $c_\tau = 0$ and $A(\mu) = \|\mu\|^q$ with $q > 1$, then $\mathcal{F}_{\mathcal{T}} = \mathcal{G}_{\mathcal{T}}$.*

The proof of this Theorem can be found in Appendix A.1.

Remark 2. *In particular with $q = 2$ we get the Gaussian loss function with $\eta_1 = 0$. Assuming $\eta_1 = 0$ is not restrictive because if η_1 is not 0 we can consider the translated data $(x_t - \eta_1)$.*

Remark 3. *A more general approach with milder conditions for function A is proposed in Appendix B. One sufficient condition consists of the strong convexity of function A with $A(0) = 0$.*

3.3 Half-space intersections and functional pruning

All g_τ functions are linear in (λ, μ). Considering yet another variable κ we can associate to every g_τ a function h_τ and a half-space H_τ of $\mathbb{R}^{p+2}$:

$$H_\tau = \{\kappa, \lambda, \mu \mid h_\tau(\kappa, \lambda, \mu) = a_\tau \lambda - 2\langle b_\tau, \mu\rangle + c_\tau - \kappa \le 0\}.$$

From here we straightforwardly get that our linearised functional problem is included in a half-space intersection problem. This result is formalised in the following theorem.

Theorem 3. *If τ is in $\mathcal{G}_\mathcal{T}$ then*

1. *there exists (κ, λ, μ) such that $h_\tau(\kappa, \lambda, \mu) = 0$ and for all $\tau' \neq \tau$, $h_{\tau'}(\kappa, \lambda, \mu) < 0$;*
2. *the hyperplane defined by $h_\tau(\kappa, \lambda, \mu) = 0$ is a supporting hyperplane of the set $\cap_\tau H_\tau$.*

Remark 4. *The set $\cap_\tau H_\tau$ defines an unbounded polyhedron as taking any $\kappa \ge \max_\tau\{c_\tau\}$, $\lambda = 0$, and $\mu = 0$ we solve all inequalities.*

In Figure 1 we illustrate our linearised functional problem for the special case where, for all τ in $\mathcal{T}$, $c_\tau = 0$.

Computing the intersection of $|\mathcal{T}|$ half-spaces is a well-studied problem in computational geometry. It is closely related to the convex hull problem by projective duality [43] and several algorithms have been proposed (see for example [44, 45, 46, 47, 48, 49]). See [50] for a recent short introduction to this topic.

3.4 Projective-duality

In this section we study the special case where for all τ in $\mathcal{T}$ $c_\tau = 0$. The following theorem states that in this case the half-space intersection problem in dimension $p + 2$ is dual to a convex hull problem in dimension $p + 1$.

Theorem 4. *If for all τ in $\mathcal{T}$, $c_\tau = 0$, then τ being in $\mathcal{G}_\mathcal{T}$ is equivalent to the point (a_τ, b_τ) in $\mathbb{R}^{p+1}$ being a vertex of the convex hull of $\{(a_\tau, b_\tau)\}_{\tau\in\mathcal{T}}$.*

The proof can be found in Appendix A.2.

We illustrate the points on the convex hull of Theorem 4 for $p = 1$ and $p = 2$ in Figure 2. As will be seen in Section 4 this is relevant for the online detection of a single changepoint (see also equation (6)). Importantly, computing the convex hull of $|\mathcal{T}|$ points is a well-studied problem in computational geometry. Over the years many algorithms have been developed [51, 52, 48, 53, 54, 55, 56]. In the rest of this paper we will only use the *QuickHull* algorithm of [48] and its implementation in the qhull library `http://www.qhull.org/`. The *QuickHull* algorithm works in any dimension and is reasonably fast for our application up to $p = 5$.

4 On The Detection of a Single Changepoint Online

4.1 From Functional Description to Convex Hull

We now apply the results of Section 3 to the problem of detecting a single changepoint described in Section 2.2, to develop an efficient online algorithm. To do this, it is helpful to define a one to one mapping P between any changepoint τ and a point in dimension $p + 1$

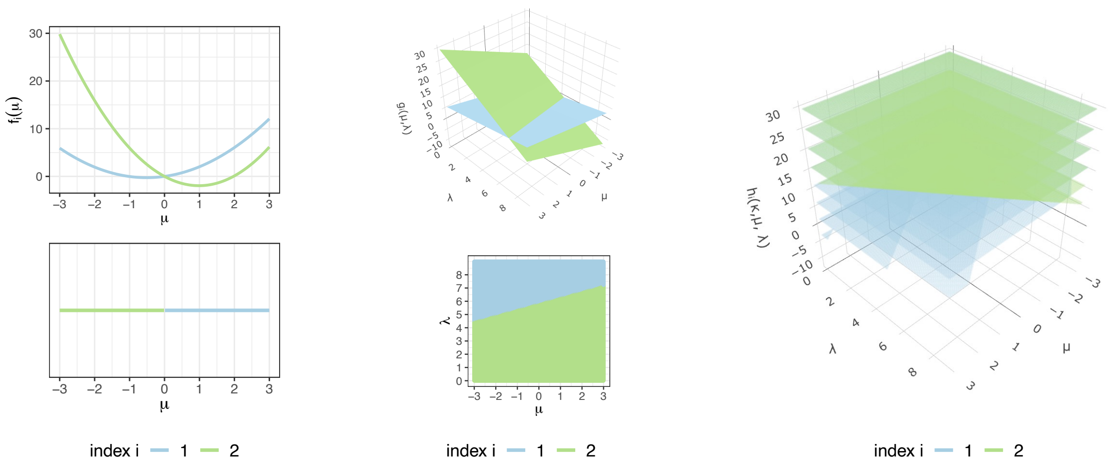
$$P(\tau) = \left(\tau, \sum_{t=1}^{\tau} x_t\right).$$

The following theorem, based on Theorem 4 essentially states that the changepoint maximising the likelihood (4) for some value η_1 and η_2 corresponds to a points on a particular convex hull.

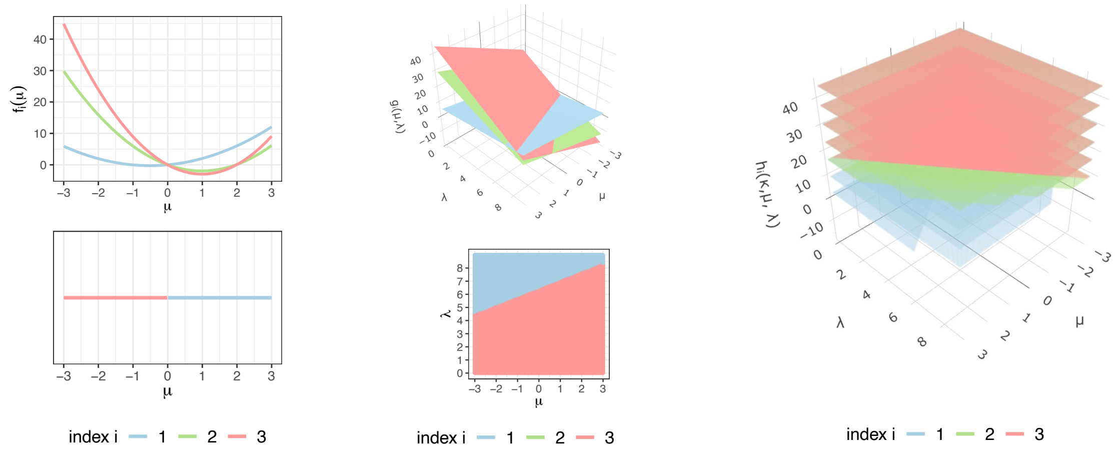
Theorem 5. *Assume the log-likelihood, $\ell_{\tau,n}(\eta_1, \eta_2)$ of a change at τ in $\{1, \dots, n-1\}$ can be written as in equation* (4)*. Then, for any η_1 taking $\hat{\tau} = \hat{\tau}(\eta_1)$ or $\hat{\tau} = \hat{\tau}(.)$ we have that $P(\hat{\tau})$ is on the convex hull of $\{P(\tau)\}_{\tau\in\{1,\dots,n-1\}}$.*

The proof of this theorem can be found in Appendix A.3.

Note that Theorem 5 is coherent with the notion of pruning as in the FOCuS algorithm [34]. If $P(\tau)$ is strictly inside the hull (i.e. not on the boundary of the hull $\{P(\tau)\}_{\tau\in\{1,\dots,n-1\}}$) at time n, then it is certainly inside the hull at any future time $n' \ge n$ (i.e. not on the boundary of the hull $\{P(\tau)\}_{\tau\in\{1,\dots,n'-1\}}$). Thus based on Theorem 5 the likelihood associated with the change at time τ will never be optimal after time n.



(a) Iteration $n = 3$



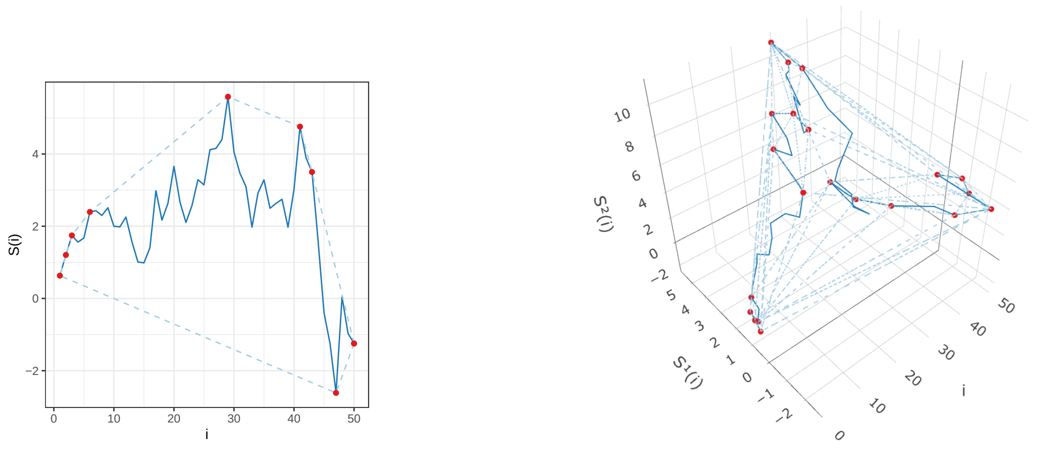
(b) Iteration $n = 4$

Figure 1: Example of the functional pruning problem and linearised functional problem over times: f_i and $\mathrm{proj}\{\max_i\{f_i(\mu)\}\}$ functions (left); g_i and $\mathrm{proj}\{\max_i\{g_i(\mu,\lambda)\}\}$ functions(middle); the level surfaces of h_i functions, where each surface is for a fixed κ value (right) for changepoint candidates $i \in \mathcal{T}$. We simulate time series $\{x_t\}_{t=1,2,..} \sim \mathcal{N}(0,1)$ and consider two time-steps, $n = 3$ and $n = 4$. We define the coefficients a_i as i, b_i as $\sum_{t=1}^{i} x_t$ and c_i as zero. Note that the change $i = 2$ associated with quadratics f_2 and surface g_2 is pruned at $n = 4$.

From a computational perspective, Theorem 5 is useful only if the set of points on the convex hull is significantly smaller than n. In the following subsections, we will prove using [57] that, under natural assumptions, the number of vertices of the convex hull of $\{P(\tau)\}_{\tau\in\{1,\dots,n-1\}}$ is of order $2(\log n)^p/p!$. This shows that for sufficiently large n the number of candidate changepoints one need to store to optimise the likelihood (or the likelihood ratio test) is much smaller than n.

4.2 Bound on The Number of Changepoint Candidates

In this subsection using the recent paper [57], we provide the expected number of faces and vertices of the convex hull $\{P(\tau)\}_{\tau\in\{1,\dots,n-1\}}$, assuming that the components of x_τ, x_τ^d (with $d = 1,\dots,p$), are independent and identically distributed with a continuous distribution. In turn, using Theorem 4 this gives us an upper bound on the expected number of changepoints that can maximise the likelihood function (4) for some value of η_1 and η_2. From a computational perspective, this bound will be useful to derive guarantees on the computational complexity of the procedure (see Algorithm 1 and 3).



(a) Dimension $p = 1$ (b) Dimension $p = 2$

Figure 2: Plot of $S(i) = \sum_{t=1}^{i} x_t$ as a function of i (dark blue) and the convex hull of $\{(a_i = i, b_i = S(i))\}_{i\in\{1,\dots,n-1\}}$ (dashed blue line) for dimensions $p = 1$ and 2 . The vertices of the convex hull $\tau \in \mathcal{T}$ are marked in red. We simulated time series $\{x_t\}_{t\in\{1,\dots,n\}}$ with $n = 51$ data points (without changepoints, $x_t \sim \mathcal{N}_p(0, I_p)$ i.i.d).

To apply the result of [57] we need to prove that the random walk $\{P(\tau)\}_{\tau\in\{1,\dots,n-1\}} = \{(\tau, \sum_{t=1}^{\tau} x_t)\}_{\tau\in\{1,\dots,n-1\}}$ satisfies two properties, (Ex) and (GP) defined below.

- (Ex) For any permutation σ of the set $\{1, \dots, n-1\}$, we have the distributional equality
$$\{(1, x_\tau)\}_{\tau\in\{1,\dots,n-1\}} \overset{D}{=} \{(1, x_{\tau'})\}_{\tau'\in\sigma}$$
- (GP) Any $p + 1$ vectors of the form $(\tau, \sum_{t=1}^{\tau} x_t)$ are almost surely linearly independent.

Lemma 1. *Assuming that all x_τ^d are independent and identically distributed with a continuous distribution, the properties (Ex) and (GP) of [57] are satisfied.*

The proof of this lemma can be found in Appendix A.4.

We can now state the expected number of faces and vertices of the convex hull of $\{P(\tau)\}_{\tau\in\{1,\dots,n-1\}}$.

Theorem 6. *Assuming all x_τ^d (with $d = 1, \dots, p$) are independent and identically distributed with a continuous distribution, then the expected number of faces (denoted as U_n^p) and vertices (denoted as V_n^p) of the convex hull of $\{P(\tau)\}_{\tau\in\{1,\dots,n-1\}}$ are the following:*

$$\mathbb{E}(U_n^p) = \frac{2p!}{(n-1)!} \begin{bmatrix} n \\ p+1 \end{bmatrix}, \quad \mathbb{E}(V_n^p) = \frac{2}{(n-1)!} \sum_{l=0}^{\infty} \begin{bmatrix} n \\ p+1-2l \end{bmatrix},$$

where $\begin{bmatrix} n \\ m \end{bmatrix}$ *are the Stirling numbers of the first kind [58].*

Proof. Based on Lemma 1 properties (Ex) and (GP) hold and we can apply formula (2) from [57] on the expected values of U_n^p and V_n^p. □

Corollary 1. *We have the following equivalents for the expected numbers of faces U_n^p and vertices V_n^p:*

$$\mathbb{E}(U_n^p) \sim 2(\log(n))^p, \quad \mathbb{E}(V_n) \sim \frac{2}{p!} (\log(n))^p, \quad n \to \infty.$$

The proof of Corollary 1 follows from Remark 1.4 of [57]. The theorem and corollary provide an expected upper limit on the number of points, or candidate locations for changepoints, that need to be evaluated to compute the CUSUM likelihood ratio test (4) exactly without resorting to approximations. Although this limit increases as $\mathcal{O}(\log(n)^p)$ and thus as a power of p, it is not overly restrictive. In some online applications, we often encounter situations where $n >> p$, making an exact implementation feasible. In the following empirical evaluations, we will evaluate scenarios up to 5 dimensions.

Remark 5. *Theorem 6 gives the expectation for i.i.d observations. One might wonder what happens if there is a changepoint in the distribution at $i < n$. Note that the points on the hull $\{P(\tau)\}_{\tau \in \{1,\dots,n-1\}}$ are necessarily on the hull of $\{P(\tau)\}_{\tau \in \{1,\dots,i\}}$ or on the hull of $\{P(\tau)\}_{\tau \in \{i+1,\dots,n-1\}}$. Thus applying the theorem on all points from 1 to i and from $i+1$ to $n-1$ we expect at most $\mathbb{E}(V^p_{i+1}) + \mathbb{E}(V^p_{n-i}) \leq 2\mathbb{E}(V^p_n)$ points on the hull. Thus if there is a changepoint (and assuming we haven't stop because the GLR statistics is lower than the specified threshold) the expected number of candidate changepoints is still in $\mathcal{O}(\log(n)^p)$.*

4.2.1 Empirical validation of the bound

We empirically tested the validity of Theorem 6. We use the Gaussian distribution as an example of a distribution with continuous density and independent dimensions. To be specific, for p from 1 to 5, we simulated time series $x_{1:n}$, with $x_t \sim \mathcal{N}_p(0, I_p)$ i.i.d. We varied the time series length from $2^{10} + 1 (\approx 10^3)$ to $2^{23} + 1 (\approx 8 \times 10^6)$ and simulated 100 data sets for each length. We used the *QuickHull* algorithm [48] implemented in the R package *geometry* [59] to evaluate the number of faces and vertices of the convex hull of $\{P((\tau)\}_{\tau \in \{1,\dots,n-1\}}$. It can be seen in Figure 3 that the observed number of faces and vertices are close to their theoretical expectations (from Theorem 6). Additional details on the calculations of the Stirling numbers are provided in Appendix C.

The proof of Theorem 6 depends on the fact that the distribution is continuous. We thus empirically tested the robustness of the expectations to this assumption by considering time series drawn with a discrete i.i.d Poisson distribution. The results closely resemble the Gaussian case and details can be found in Appendix E.3.

4.3 A simple online algorithm

In this section we propose an algorithm to compute the GLR statistics at every time step n for a known and unknown η_1. In both cases, based on Theorem 5 we only need to consider the changepoints τ such that $P(\tau)$ is on the hull of $\{P(\tau)\}_{\tau \in \{1,\dots,n-1\}}$.

Let us call $\mathcal{T}_n$ the set of index τ such that $P(\tau) = (\tau, \sum_{t=1}^{\tau} x_t)$ is on the convex hull of all $\{P(\tau)\}_{\tau \in \{1,\dots,n-1\}}$. Restating Theorem 4, the changepoint $\hat{\tau}$ maximising the likelihood (4) (i.e $\hat{\tau}(\eta_1)$ if η_1 is known or $\hat{\tau}(.)$ if it is not) is in $\mathcal{T}_n$. Hence, computationally speaking, at step n we only need to compute the likelihood (and likelihood ratio statistics) of changepoints τ in $\mathcal{T}_n$. Furthermore, as already explained after Theorem 5 we can prune all τ that are not in $\mathcal{T}_n$ as they will never be in $\mathcal{T}_{n'}$ for any $n' \geq n$.

Computing the convex hull from scratch at every-time step n is inefficient: it would require at least $\mathcal{O}(n)$ operations and lead to an overall complexity at least quadratic in the number of data points. Ideally, we would like to update the hull using an efficient online or dynamic convex hull algorithm as the one in [60]. There exists implementations of such an algorithm in dimension $p = 2$ [61] and $p = 3$ [62] However, out of simplicity, here we rely on an iterative use of the offline *QuickHull* algorithm [48] that also works for larger dimensions and is available in the geometry R package [59].

In details, for any $n' > n$ we have $\mathcal{T}_{n'} \subset \mathcal{T}_n \cup \{n, \dots, n'-1\}$. Therefore if we already computed $\mathcal{T}_n$, with *QuickHull*, when we compute $\mathcal{T}_{n'}$ we need not restart from scratch but simply apply *QuickHull* to the points $P(\tau)$ with τ in $\mathcal{T}_n \cup \{n, \dots, n'-1\}$. A first idea would be to apply the *QuickHull* algorithm at every time step $n, n+1, \dots$, that is compute $\mathcal{T}_n$ as the hull of all $P(\tau)$ with τ in $\mathcal{T}_{n-1} \cup \{n\}$. However, as seen in Figure 10, when computing the hull too often, the overhead of *QuickHull* significantly slows the overall algorithm, even in low dimensions.

In practice, we discovered that running the *QuickHull* algorithm intermittently, rather than at every iteration, resulted in faster run times. Specifically, we maintain a set of linearly increasing indices, denoted as $\mathcal{T}'_n$, and only periodically execute *QuickHull* to reconstruct the true $\mathcal{T}_n$. More precisely, to decide when to run *QuickHull* we update a maximum size variable for $\mathcal{T}'_n$ (denoted $maxSize$). Then, at step $n+1$

- either the size of $\mathcal{T}'_n$, $|\mathcal{T}'_n|$, is smaller than $maxSize$ in that case we set $\mathcal{T}'_{n+1}$ to $\mathcal{T}'_n \cup \{n\}$,
- or it is larger and in that case we compute $\mathcal{T}'_{n+1}$ as the convex hull of all $P(\tau)$ with τ in $\mathcal{T}'_n \cup \{n\}$.

In the later case we also update $maxSize$ to $maxSize = \lfloor \alpha |\mathcal{T}'_{n+1}| + \beta \rfloor$, with $\alpha \geq 1$ and $\beta \geq 0$. It is important to note that, in this way, we always retain a few more possible changepoints than necessary: $\mathcal{T}_{n+1} \subseteq \mathcal{T}'_{n+1}$.

Remark 6. *Taking $\beta = 1$ and $\alpha = 1$, we run the QuickHull algorithm at every time-step. Increasing the value of α allows to run it less often at the cost of having a longer list of indices $\mathcal{T}'_n$. In Figure 10 we considered different value for α with $\beta = 1$ and found that a value of $\alpha = 2$ gives good practical performances (see Appendix E.1).*

A formal description of our procedure, called MdFOCuS if η_1 is unknown and MdFOCuS0 if it is, is available in Algorithm 1.

Algorithm 1 MdFOCuS(0) algorithm

```
1: Input 1: $p$-variate independent time series $\{x_t\}_{t=1,2,...}$
2: Input 2: threshold $thrs$ and $\eta_1$ if known pre-change parameter
3: Input 3: pruning index $maxSize > P + 1$, the parameters $\alpha \geq 1$, $\beta \geq 0$.
4: Output: stopping time $n$ and maximum likelihood change : $\hat{\tau}(.)$ or $\hat{\tau}(\eta_1)$
5: $likelihoodRatio \leftarrow -\infty, \quad \mathcal{T} \leftarrow \emptyset, \quad n \leftarrow 0$
6: while ($likelihoodRatio < thrs$) do
7:     $n \leftarrow n + 1$
8:     $\mathcal{T} \leftarrow \mathcal{T} \cup \{n - 1\}$
9:     $maxLikelihood \leftarrow \max_{\tau \in \mathcal{T}}\{\max_{\eta_1,\eta_2} l_{\tau,n}(\eta_1, \eta_2)\}$          ▷ $\eta_1$ is fixed in MdFOCuS0
10:    $likelihoodRatio \leftarrow maxLikelihood - \max_{\eta_1,\eta_2} l_{n,n}(\eta_1, \eta_2)$          ▷ $\eta_1$ is fixed in MdFOCuS0
11:    if ($|\mathcal{T}| > maxSize$) then
12:        $\mathcal{T} \leftarrow$ QUICKHULL($\{P(\tau)\}_{\tau \in \mathcal{T}}$)          ▷ Returns the index on the hull of all $P(\tau)$.
13:        $maxSize \leftarrow \lfloor \alpha|\mathcal{T}| + \beta \rfloor$
14:    end if
15: end while
16: return $n$ and $likelihoodRatio$
```

The only difference between MdFOCuS and MdFOCuS0 variants are in the calculation of the likelihood (line 9 and 10). Therefore the two variants store the same number of candidate changepoints and have the same time and memory complexity.

We give a link to an implementation of Algorithm 1 for a change in the mean of p-dimensional Gaussian signal in about a hundred lines of R code in the Supplementary Material. Our code is based on the *convhulln* function of the *geometry* R package [59] and for $p = 3$ it is able to process in 2 minutes $n = 5 \times 10^5$ data points on an Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz processor choosing $\alpha = 2$ and $\beta = 1$. As will be seen in Section 4.3.1 and Figure 4 its implementation in R/C++ is even faster. Note that such a run time is comparable to that of the ocd package [37] as for the same simulated data and on the same processor, ocd executes on average in 3 minutes and 18 seconds.

Algorithm 1 is exact and empirically fast but to the best of our efforts, it is not trivial, to derive its worst or expected time complexity. In Appendix F we propose a modified algorithm, updating the set $\mathcal{T}_n$ (line 11 to 14) in a deterministic manner and prove that it is quasi-linear on average, with an $\mathcal{O}(n \log(n)^{p+1})$ complexity, under the assumption that the convex hull algorithm we are using is at worst quadratic. Empirically, this algorithm is harder to implement and slower than Algorithm 1 (see Figure 15 in Appendix F). The assumption on the quadratic complexity of the convex hull algorithm is valid for QUICKHULL up to dimension $p = 2$ [48, 63] and using the algorithm of [64] up to dimension $p = 3$.

4.3.1 Additional Empirical Evaluations

We implemented Algorithm 1 for a change in the mean of multi-dimensional Gaussian signal (see Table 1) in R/C++ using the *qhull* library (see link in the Supplementary Material). We evaluated its running time. We simulated data as in the empirical validation of Theorem 6 on the expected number of faces and vertices for dimension $p = 1$ to $p = 5$. The average run times as a function of n are presented in log-scale in Figure 4. As expected, it is faster than our R implementation: for example for $p = 3$ it process in 2 minutes $n = 10^6$ rather than 5×10^5. Note also that the run times for a known or unknown η_1 are almost identical. Finally, we made a simple linear regression to model the logarithm of the run time as a function of $\log(n)$. The estimated slope coefficients are reported in Figure 4 and are all smaller than 1.25.

4.3.2 Simulation Study

We evaluate empirical performances of the R implementation of MdFOCuS on synthetic data (a link to the code is available in the Supplementary Material). Specifically, we analyse 3-dimensional time series under the Gaussian change-in-mean setting. We consider the two scenarios of pre-change mean known and unknown. For each experiment, we report results aggregated from 300 data sets. A similar study on 5-dimensional time series can be found in Appendix E.2.

We compare performances of MdFOCuS with the high-dimensional online changepoint detection method ocd from [37], and a naive multivariate implementation of FOCuS, obtained by running FOCuS independently for each dimension and both summing and taking the maximum of the resulting traces (see Section 6 of [34]).

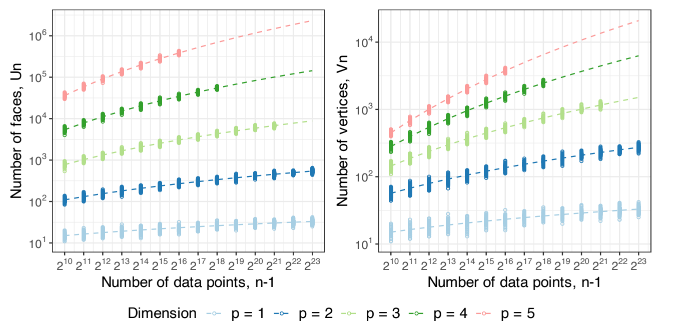


Figure 3: The number of faces (left) and vertices (right) of the convex hull of $\{P(\tau)\}_{\tau\in\{1\dots,n-1\}}$ for dimension $1 \leq p \leq 5$. We simulated 100 i.i.d. Gaussian data $\mathcal{N}_p(0, I_p)$ for n from $(2^{10} + 1)$ to $(2^{23} + 1)$. Dashed lines correspond to the expected number of faces (left) and vertices (right) presented in Theorem 6. We consider a maximum running time of 10 minutes for the *QuickHull* algorithm. This is why some results for $p = 3, 4$ and 5 for large signals are missing.

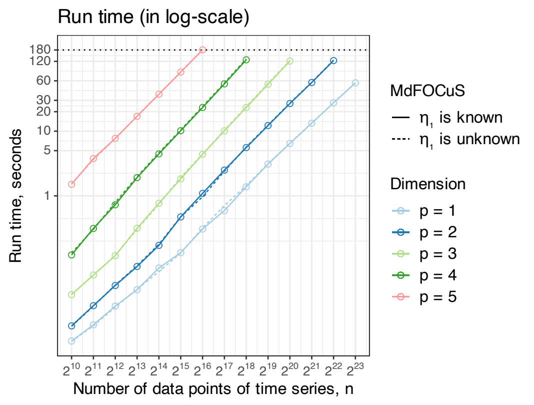


Figure 4: Run times in seconds of MdFOCuS with known (dashed line) and unknown(full line) pre-change parameter η_1, both with $\alpha = 2$ and $\beta = 1$, in dimension $p = 1, \dots, 5$ using time series $x_{1:n}$ (without change, $x_t \sim \mathcal{N}(0, I_p)$ i.i.d). Run times are averaged over 100 data sets. We considered a maximum running time of 3 minutes (horizontal dotted black line) for the algorithms. That is why we do not report run times for $p = 3, 4$ and 5 for large signals. We report here the slope of the curves estimated using a simple linear regression: for known η_1: ≈ 1.04 ($p = 1$), ≈ 1.16 ($p = 2$), ≈ 1.22 ($p = 3$), ≈ 1.24 ($p = 4$) and ≈ 1.14 ($p = 5$); for unknown η_1: ≈ 1.05 ($p = 1$), ≈ 1.16 ($p = 2$), ≈ 1.23 ($p = 3$), ≈ 1.24 ($p = 4$) and ≈ 1.14 ($p = 5$).

Thresholds were selected under a Monte-Carlo simulation to achieve an average run length of 5×10^3 observations, in line with the empirical evaluation studies of both of [34, 37]. For a change at $t = 500$, we check performances for a range of different change magnitudes and affecting a different numbers of time series. In order to compare performances across different sparsity regimes, changes in the affected time series were normalized in such a way that the change direction lies in the p-dimensional sphere, e.g. ($||\eta_1 - \eta_2||_2^2 = \delta$), where δ is the overall change magnitude and η_1, η_2 are respectively the pre- and post-change mean.

Starting with the pre-change mean known scenario, in Tables 2 and 3 we report respectively the average detection delay and the standard deviation of the detection delay over various magnitudes and for different change densities. With FOCuS0, MdFOCuS0 and ocd (oracle) we indicate the respective algorithms with oracle pre-changepoint parameter $\eta_1 = 0$.

Compared to other algorithms, MdFOCuS0 tends to achieve smaller detection delays in those cases where the change affects all the time series, in particular over changes of low magnitude. In addition, MdFOCuS0 tends to have more consistent performances across different sparsity regimes within the same change magnitudes, as in general it shows lower variance in the estimates of the standard deviation of the detection delays.

As above, we report in Tables 4 and 5 the average and standard deviation of the detection delay in the pre-change mean unknown scenario. In addition to the methods that naturally extend to the pre-change mean unknown case (FOCuS and MdFOCuS), we consider ocd and FOCuS0 with pre-change mean η_1 known and estimated from a probation period of 500 observations. These observations are not from the test set, and were considered as an additional training set. We denote these two algorithms as FOCuS0 (est) and ocd (est).

We observe similar results to the pre-change mean known case, where MdFOCuS outperforms competitor methods over the dense cases in terms of detection delay, and shows lower variability across the various estimates across most of the change magnitudes.

Table 2: Average detection delay in the pre-change mean known scenario with 3-dimensional time series for a change at time $t = 500$. Best results per row are highlighted in bold.

magnitude δ	density	FOCuS0	MdFOCuS0	ocd (oracle)
0.25	1	250.30	272.48	**230.63**
0.25	2	264.31	266.93	**254.48**
0.25	3	281.10	**272.61**	292.61
0.50	1	69.56	73.18	**59.64**
0.50	2	74.59	69.85	**66.27**
0.50	3	77.71	**71.80**	72.62
1.00	1	**18.68**	19.81	18.95
1.00	2	20.84	**19.59**	20.32
1.00	3	22.18	**19.98**	21.84

Table 3: Standard deviation of the detection delay in the pre-change mean known scenario with 3-dimensional time series for a change at time $t = 500$. Best results per row are highlighted in bold.

magnitude δ	FOCuS0	MdFOCuS0	ocd(oracle)
0.25	4.57	**1.05**	23.45
0.50	**0.83**	1.54	3.36
1.00	0.79	**0.31**	0.99

Table 4: Average detection delay in the pre-change mean unknown scenario with 3-dimensional time series for a change at time $t = 500$. Best results per row are highlighted in bold.

magnitude δ	density	FOCuS	FOCuS0 (est)	MdFOCuS	ocd (est)
0.50	1	81.67	84.57	85.83	**74.26**
0.50	2	92.07	99.32	85.01	**84.57**
0.50	3	96.08	106.51	**86.88**	91.48
1.00	1	**19.89**	22.34	20.51	22.18
1.00	2	22.38	25.67	**20.41**	23.51
1.00	3	23.28	28.31	**20.30**	24.71
2.00	1	**5.53**	6.22	5.78	7.04
2.00	2	6.19	7.41	**5.76**	7.39
2.00	3	6.48	8.00	**5.78**	7.50

Table 5: Standard deviation of the detection delay in the pre-change mean unknown scenario with 3-dimensional time series for a change at time $t = 500$. Best results per row are highlighted in bold.

magnitude δ	FOCuS	FOCuS0 (est)	MdFOCuS	ocd (est))
0.50	4.18	4.56	**1.24**	3.49
1.00	**0.51**	0.80	0.52	0.61
2.00	0.24	0.40	**0.02**	0.10

5 Application on NBA Plus-Minus

In this section, we illustrate our methodology to detect a change in the Cleveland Cavaliers (an NBA team) Plus-Minus score from season 2010-11 to 2017-18. This example was proposed in [38]. We use this example for two reasons. First, it highlights that our computational framework is also applicable to CUSUM Exponential baseline e-detectors proposed in [38] which in essence are a non parametric generalization of generalized likelihood ratio rules. Second, it illustrates how modelling not only the signal's mean but also its variance can simplify the calibration of the detection threshold, without necessarily resorting to a non-parametric approach.

We recovered the Plus-Minus score using the nbastatR package [65] available on GitHub `https://github.com/abresler/nbastatR`. The Cavaliers Plus-Minus is represented in Figure 5. This score represents how well the Cavaliers fared against their opponent. In this subsection, we assume that we are observing these matches one after the other and we are interested in detecting a changepoint online.

As detailed in [38] it is nontrivial to fit this problem into commonly used parametric sequential changepoint detection procedures for two main reasons. First, it is not easy to choose a parametric model to fit the score. In particular we

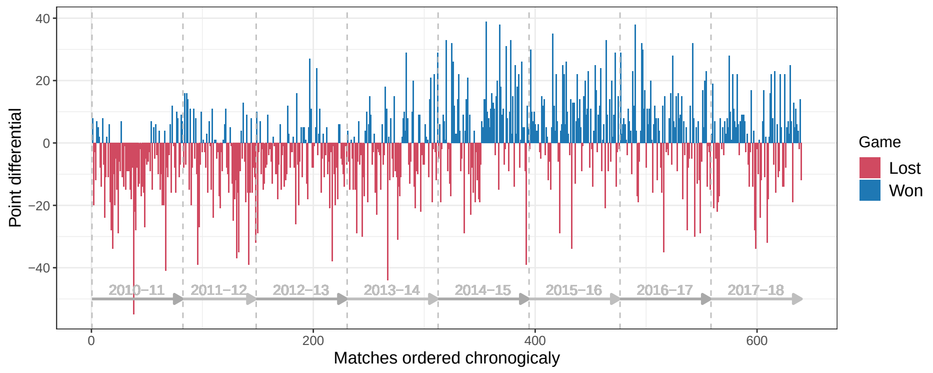


Figure 5: Plus-Minus score of the Cavaliers from season 2010-11 to season 2017-18 as a bar plot. Losses are in light blue and wins are in dark blue. Ends of seasons are represented with vertical dashed lines and seasons are shown on the bottom with grey arrows.

would like to take into account that we have integer-valued data, and we expect some form of dependency. Second, assuming we have chosen a model or likelihood function, calibrating the threshold to detect a changepoint is not trivial. A small change in the distribution we use as a reference can substantially change the threshold and as a consequence increase the detection time or lead to false detections. E-detectors of [38], are an interesting non-parametric solution to these two problems assuming some knowledge of the pre-change distribution.

5.1 Convex hull for CUSUM e-detectors

Following equations (27) of [38] and considering an exponential baseline increment as in their equation (41) we can write their update of the e-detector CUSUM statistics for a given λ as

$$M_n^{CU}(\lambda) = \exp\left(\lambda s(x_n) - \psi(\lambda) v(x_n)\right) \max\{M_{n-1}^{CU}(\lambda), 1\},$$

where λ is in a subset of $\mathbb{R}$, s and v are real-valued functions, and ψ is a finite and strictly convex function. Taking the $\log$ and considering all possible changepoints prior to n we recover with induction that

$$\log(M_n^{CU}(\lambda)) = \max_{\tau<n} \left\{ \sum_{t=\tau+1}^{n} (\lambda s(x_t) - \psi(\lambda) v(x_t)) \right\}.$$

Subtracting $\sum_{t=1}^{n}(s(x_t)\lambda - \psi(\lambda)v(x_t))$, we recover our functional pruning framework of equation (7) taking for any changepoint at $\tau < n$, $a_\tau = \sum_{t=1}^{\tau} v(x_t)$, $2b_\tau = \sum_{t=1}^{\tau} s(x_t)$, and $c_\tau = 0$. Applying Theorem 4 and its proof we get that the changepoint optimising the e-detector CUSUM score for a given λ value corresponds to a vertex on the convex hull of the following points in $\mathbb{R}^2$:

$$\left(\sum_{t=1}^{\tau} v(x_t), \frac{1}{2} \sum_{t=1}^{\tau} s(x_t) \right)_{\tau<n}. \tag{9}$$

As argued in [38] it makes sense to consider a mixture of e-detectors, that is different λ values. For computational reasons [38] considered a finite number of values. In practise, this number is determined using their computeBaseLine function that is detailed in their appendix B in Algorithm 3.

We will now compare the baseline numbers of two e-detectors for Plus-Minus score to the number of points on the hull of (9). [38] considered two e-detector models. In the first, that we call Winning rate, they model the winning rate and consider, see their equation (64), for the exponential increment $v(x_t) = 1$ and $s(x_t) = \mathbb{1}_{x_t>0} - p_0$, where $\mathbb{1}$ is the indicator function and p_0 is taken to be 0.49 in the numerical application. With the function computeBaseLine they report selecting 69 λ values. In their second e-detector model, that we call Plus-Minus, they analyse the Plus-Minus score more directly. The score is first normalized between 0 and 1 as follows $x_t' = (x_t + 80)/160$. Then they define $s(x') = (x'/m - 1)$ and $v(x) = (x'/m - 1)^2$, where m is chosen to be 0.494 in the numerical application. With the computeBaseLine function they report selecting 190 λ values. For both e-detectors we computed the number of points in the convex hull for n between 10 and $n = 640$ (the last match of season 2017-18) and obtained for Winning Rate at most 16 and for Plus-Minus at most 18 points. Therefore out of the, respectively, 69 and 190 candidates suggested by computeBaseLine many are redundant as they correspond to the same changepoint.

5.2 A parametric approach to the Plus-Minus score

Considering a change in the mean of Gaussian model to fit the Plus-Minus score of the Cavaliers seems rather naive, as it does not take into account the discrete nature of the data and possible dependencies. In this context we argue that it is also not clear how one could get a valid estimation of the pre-changepoint parameter. Finally, as argued in [38] and shown a bit later calibrating the threshold to detect a changepoint is not trivial. Roughly speaking the threshold depends on the choice of the null distribution and this largely affect the position at which a change is detected. Here as a robust parametric solution to these problems we consider a change in the mean and variance of a univariate Gaussian signal with unknown pre-changepoint parameters.

For a change in mean and variance using our convex hull framework equation (7) and following its natural parametrisation presented in Section 2.2 we need to keep track of the vertices on the hull of a 3-dimensional set of points

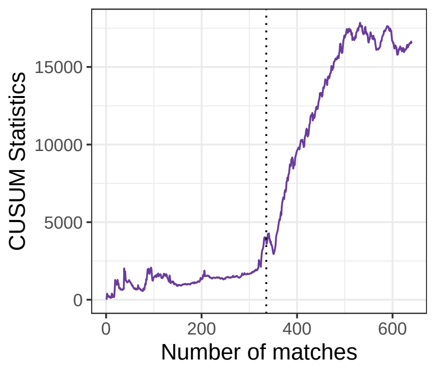
$$\left(\tau, \sum_{t=1}^{\tau} x_t, \sum_{t=1}^{\tau} x_t^2\right)_{\tau<n}. \tag{10}$$

As we will see this second model, although still parametric, is much more robust to the choice of the null distribution.

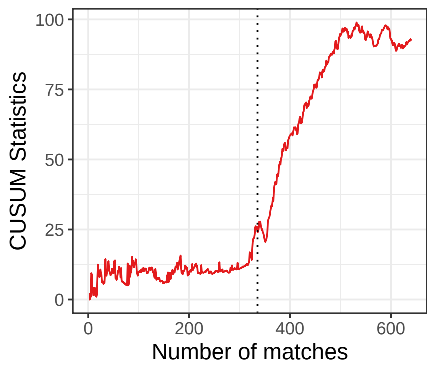
Interestingly, this mean and variance model is invariant to shift and scaling. Based on that, we see that all the points on the convex hull of the Plus-Minus e-detector are on the hull of the change in mean and variance model. Indeed for this second e-detector the raw Plus-Minus score was normalized as follow $x'_t = (x_t + 80)/160$ and then s and v were chosen to be $s(x') = (x'/m - 1)$ and $v(x') = s(x')^2)$. Therefore all changepoints tested by the Plus-Minus e-detector will also be tested by our mean and variance model.

As the data is discrete it is not unlikely that two consecutive scores are exactly equal (this happens 19 times out of 639 in the Plus-Minus score of the Cavaliers from 2010 to 2018). In fact we also observed 2 stretches of 3 equal scores. For such stretches the estimated variance would be 0 and the likelihood would blow up to infinity. To get around this problem we considered a minimum value for the variance. We computed the per season variance of all NBA teams from season 2003-2004 to season 2018-2019 and observed a minimum of 97.02, a maximum of 247.38 and a mean of 170.68. Hence, we argue that setting a minimum variance of 1 (which is roughly a hundred times smaller) should not hamper our ability to detect changes, while also limiting the impact of small segments with equal values.

In Figure 6 we represent the CUSUM statistics for the change in mean and change in mean and variance model. Although the scales are quite different, the two statistics have very similar trends and importantly both seems to increase around season 2013-14 (starting at match number 312 represented by a vertical dotted line). Although both models reconstruct the same changepoint, we will see in the next subsection that the mean-variance model shows less variability in terms of detection delay, achieving faster detections once a threshold is set to achieve a fixed run length.



(a) Mean model



(b) Mean and variance model

Figure 6: CUSUM statistics for the mean model for the mean and variance model. The dotted line corresponds to the beginning of season 2014-15 and match number 312.

Interestingly, at any time-point after 320 (8th match of season 2014-15) the maximum likelihood changepoint is always 279 (5th of February 2014) for both models (mean and, mean and variance). Furthermore, the convex hull for the mean and variance model (in equation (10)) - and hence the mean model and Plus-Minus e-detector (recall that all changepoints tested by the Plus-Minus e-detector are tested by the mean and variance model) - no longer contains vertex 312 after time $n = 320$ (8th match of season 2014-15).

Match 279 was played on the 5th of February 2014, well before the beginning of season 2014-15 in October 2014. At first this might come as a surprise. But, it indeed seems that the mean Plus-Minus score increased after the 6th of February 2014 (average score of -6.28 before compared to 1.12 after and significant t-test p-values of 0.01584). See also the zoom on the Plus-Minus score over season 2013-14 and 2014-15 in Figure 7. Also, as discussed on the wikipedia web page of David Griffin [66] "On February 6, 2014, he was named the acting general manager for the Cavaliers" and David Griffin, was the one "to signed LeBron James back" for season 2014-15.

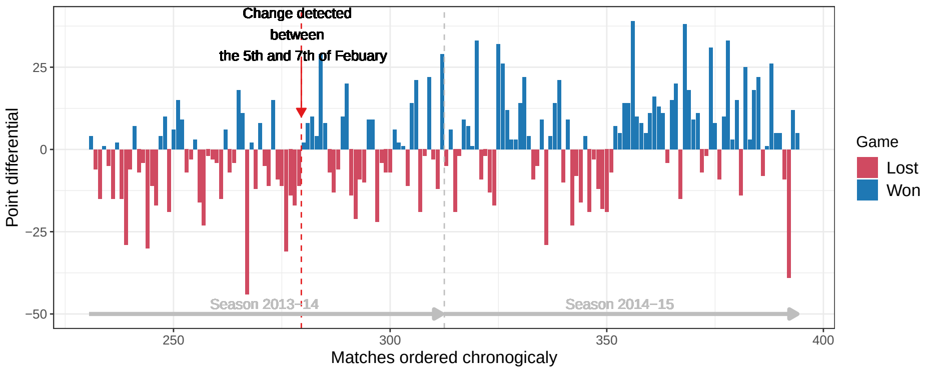


Figure 7: Plus-Minus score of the Cavaliers from season 2013-14 to 2014-15 as a bar plot. Losses are in light blue and wins are in dark blue. Ends of seasons are represented with vertical dashed lines (grey) and seasons are shown on the bottom with grey arrows.

5.2.1 Simulating NBA Plus-Minus scores without changepoint

Based on Figure 6 it is clear that for a particular range of the threshold both the mean and mean and variance model with unknown pre-changepoint parameter can detect a changepoint after the beginning of season 2014-15. To calibrate this threshold one typically need a simulation model without a changepoint. Here, we will set the CUSUM threshold such that under our null simulation over 1000 matches we would detect a change only in 5% of the cases.

Our idea here is not to precisely model the Plus-Minus score of the Cavaliers but rather explore a few natural optimistic and pessimistic options to model it and check how this affects the threshold of the two models and in turns the time at which they detect the change.

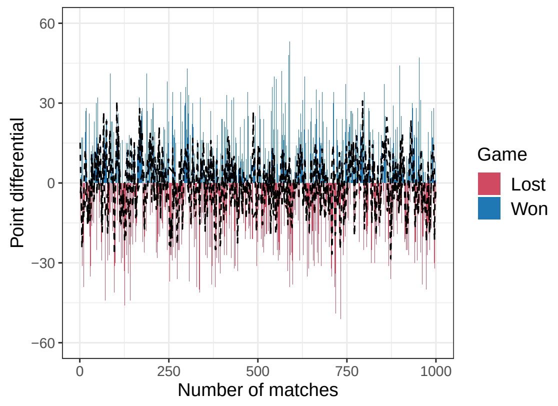
The Plus-Minus score is integer-valued. A straightforward, albeit maybe simple, choice would be to consider the difference of two independent Poisson distributions. This is the so-called Skellam distribution. It has two parameters (the mean of the first and second Poisson λ_1 and λ_2). We simulated data using values of $\lambda_1 = \lambda_2$ both equal to 95, 105, and 115. We considered mean values around 100 as the points per team of an NBA match from season 1999-00 to season 2021-22 range from 93.40 to 112.09 with a mean of 101.07.

A slightly more complex choice is to consider the difference of two independent Negative Binomial distributions with equal scale parameters, r_1 and r_2. As for the Poisson we considered $\lambda_1 = \lambda_2$, which were set to 95, 105, and 115, along with scale parameters $r_1 = r_2$ at values of 500, 100, 50, and 10. This makes a total of 12 combinations.

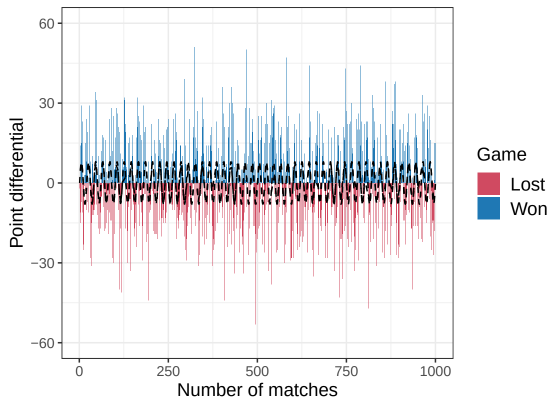
A data-driven way to model the Plus-Minus score without changepoint is to re-sample past games. We did this sampling uniformly at random past matches of the Cavaliers from season 1999-2000 to 2009-2010 or uniformly at random past matches of any NBA team from season 1999-2000 to 2009-2010. Finally, following the argument of [38] and noting that the Plus-Minus score is never larger than 80 we simulated integer valued score uniformly at random between -80 and 80.

In the Poisson and Negative Binomial based model, we could further model dependencies in the data. To do this we varied the mean of the first Poisson (or Negative Binomial) over time. We considered two scenarios. In the first we added to the mean an auto-regressive process of order 1 and coefficient 0.6 multiplied by either 1, 2, 4 and 8. In the second we added a sin waves with a period of 20 matches multiplied by either 1, 2, 4 or 8. We present in Figure 8 two simulated score profiles.

Figure 9a represent the various thresholds (in log-scale) we get for our two models, mean and mean and variance. The corresponding detection times are represented in Figure 9b.



(a) Using AR(1) with AR parameter 0.6 (multiplied by 8)

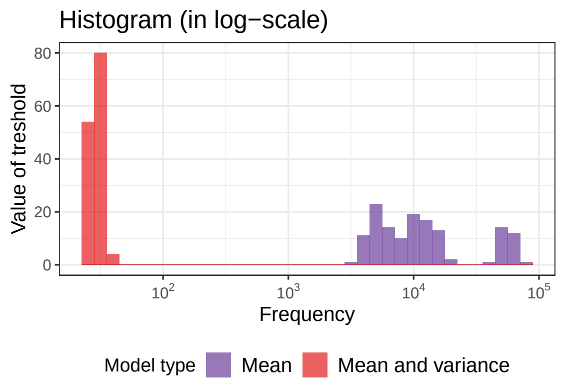


(b) Using sin waves with a period of 20 matches (multiplied by 8)

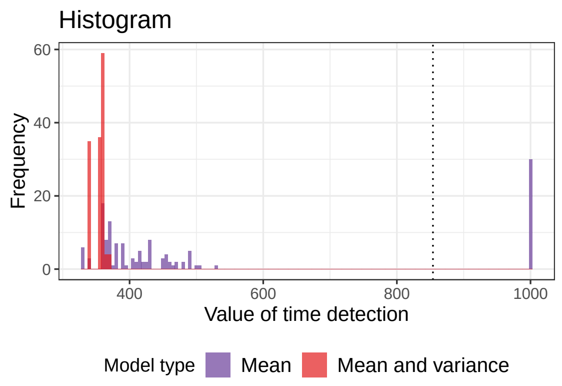
Figure 8: Two Plus-Minus score simulation examples based on the Negative Binomial model with a mean of 115 and scale parameter of 50 using two dependency scenarios.

It can be noted that the threshold is much more variable for the changepoint in mean model and this translates to a much more variable detection time. To be specific for the mean model, the change is detected at the earliest at match 330 (the 5th of December 2014), but in 21.7% of the cases the threshold is so high that the changepoint is never detected (reported in the histogram as a detection at 1000). For the mean and variance model, the change is always detected at the earliest at match 339 (the 23rd of December 2014) and at the latest at match 368 (the 20th of February 2015). In fact in 94.2% of the simulation models the change is detected before with the mean and variance model.

Based on this numerical application we argue that being able to run efficiently a bi-parametric model online can be beneficial even if the model (Gaussian here) does not properly fit the nature (discrete here) of the data.



(a) Threshold



(b) Detection time (1000 = not detected)

Figure 9: Threshold and Detection time obtained for the mean and mean and variance model.

6 Discussion

In this paper we establish a connection between the detection of a single changepoint and a convex hull problem in dimension $p + 1$. Assuming independence of all observations and dimensions and a continuous distribution we demonstrate using the recent probabilistic paper [57] that the expected number of faces and vertices of the convex hull is of order $2(\log(n))^p$ and $2(\log(n))^p/p!$, respectively, where n is the number of data points.

Based on this, we derive a simple yet efficient algorithm, MdFOCuS, that allows maximising the CUSUM statistics over all possible window sizes. MdFOCuS generalises the FOCuS algorithm [34] to the multivariate case. We show that MdFOCuS is empirically fast for up to $p = 5$ and allows to process more than 10^5 data points in a matter of minutes. We also show using simulation that MdFOCuS is statistically competitive in particular when the change affects all dimensions. Finally, in Appendix F, we also present a variant of MdFOCuS, which although empirically slower, has a run time which is provably quasi-linear, $\mathcal{O}(n\log(n)^{p+1})$.

A few improvements to the computational complexity of the procedure could be made to allow the procedure to scale to higher dimensions and longer sequences. The most computationally expensive components of the procedure are respectively the *QuickHull* algorithm, for pruning the list of indices, and the maximisation step. We mentioned already how further research could go towards a sequential update of the boundary of the convex hull of our points, to improve on the $\mathcal{O}(\mathcal{T})$ complexity of *QuickHull*. The idea would be to employ information from the hull at the previous iteration, evaluate inclusion of the most-recent point, and update the faces of neighbouring vertex consequently.

Recently, [67] offered a solution to improve on the maximisation step. Instead of looking at the maxima of all indices to calculate the test statistic, the authors exploit a bound on the maxima of the differences of contiguous indices. The main idea is that the overall maximum, our test statistics, is bounded by the sum of those maxima of the differences. If this sum does not pass the threshold there is no need to compute the likelihood ratio-statistics for all unpruned changepoints. Fast sequential updates of the sum of the maxima of the difference exists, effectively rendering the univariate algorithm constant per iteration [67]. Whilst this bound translates exactly to the multivariate case, it is easily much larger than the threshold, given that we need to store and sum a large amount of indices even in dimension 2. A less conservative bound could be object of future study to improve on the maximisation step in higher dimensions.

To conclude, our mathematical argument is valid for a large collection of parametric online single changepoint models with distributions taken from the exponential family. We also show in Appendix D that there is a connection between multiple changepoint models and the half-space intersection problem of Theorem 3. Furthermore we show in Section 5 that Theorem 5 could be beneficial to speed the calculation of the non-parametric e-detectors. Finally, note that no assumptions are made on the set of indices $\mathcal{T}$ in equation (7) and for example our theory also applies to changepoint models with observations on a tree rather than a chain [68]. For all these reasons we argue that this connection between changepoints and convex hull (or half-space intersection) sheds light on the geometric nature of the changepoint detection problem and is a promising line for future research.

Supplementary Material

The implementation of the MdFOCuS algorithm, the Plus-Minus score data, the computer code for the simulation study, and the applications discussed in this article can be found in the following GitHub repositories:

- `https://github.com/guillemr/Focused/tree/main` is a link to the R implementation of the MdFOCuS algorithm and R code for simulation studies.
- `https://github.com/lpishchagina/focus` is a link to the R/C++ implementation of the MdFOCuS algorithm.
- `https://github.com/lpishchagina/dyadicMdFOCuS` is a link to the R/C++ implementation of the Md-FOCuS algorithm with dyadic update. It is a different repository from the MdFOCuS algorithm because the dyadic version is interesting mostly from a theoretical perspective to bound the expected complexity but empirically much slower than MdFOCuS;
- `https://github.com/abresler/nbastatR` is the R package [65] we used to recover The Plus-Minus score data.

A Proofs

A.1 The proof of Theorem 2

Proof. Consider that τ is in $\mathcal{G}_{\mathcal{T}}$. By definition, there exists $(\lambda_0, \mu_0) \in Im(A) \times \mathcal{D}_A = \mathbb{R}^+ \times \mathbb{R}^p$ such that for all $\tau' \neq \tau$ in $\mathcal{T}$ we have: $g_\tau(\lambda_0, \mu_0) > g_{\tau'}(\lambda_0, \mu_0)$. Using condition $c_\tau = c_{\tau'} = 0$ and multiplying by $\alpha > 0$ we get by linearity

$$g_\tau(\alpha\lambda_0, \alpha\mu_0) = \alpha g_\tau(\lambda_0, \mu_0) > \alpha g_{\tau'}(\lambda_0, \mu_0) = g_{\tau'}(\alpha\lambda_0, \alpha\mu_0).$$

Therefore, the optimally of index τ is true on the half-line $\{\alpha(\lambda_0, \mu_0), \alpha > 0\}$. Note that we can exclude the cases $\|\mu_0\| = 0$ or $\lambda_0 = 0$ as, by a continuity argument, the optimal set for index τ in $\mathcal{G}_{\mathcal{T}}$ (if non empty) is an open set.

We now search for $\alpha_0 \in (0, +\infty)$ such that for any index $\tau'' \in \mathcal{T}$, $g_{\tau''}(\alpha_0\lambda_0, \alpha_0\mu_0) = f_{\tau''}(\alpha_0\mu_0)$. If such α_0 exists, we have

$$a_{\tau''}\alpha_0\lambda_0 - 2\langle \alpha_0\mu_0, b_{\tau''}\rangle = a_{\tau''}\|\alpha_0\mu_0\|^q - 2\langle \alpha_0\mu_0, b_{\tau''}\rangle\,,$$

leading to solution $\alpha_0 = \frac{\lambda_0^{\frac{1}{q-1}}}{\|\mu_0\|^{\frac{q}{q-1}}}$. α_0 is well defined as we excluded $\|\mu_0\| = 0$ and $\lambda_0 = 0$. Further note that the result is still true for $a_\tau = 0$ or $a_{\tau'} = 0$. □

A.2 The proof of Theorem 4

Proof. Consider any τ in $\mathcal{G}_{\mathcal{T}}$. Using Theorem 3 we have that $h_\tau(\kappa, \lambda, \mu) = 0$ and for all $\tau' \neq \tau$ $h_{\tau'}(\kappa, \lambda, \mu) < 0$. It can not be that $\lambda = 0$ and $\mu = 0$ as in that case $h_\tau(\kappa, 0, 0) = 0$ and then $\kappa = 0$, and $h_{\tau'}(\kappa, 0, 0) = 0$ contradicting $h_{\tau'}(\kappa, \lambda, \mu) < 0$.

Now we have the following equivalence:

1. $\exists \kappa, \lambda, \mu,$ with $(\lambda, \mu) \neq 0$ such that $h_\tau(\kappa, \lambda, \mu) = 0$ and $\forall \tau' \neq \tau \quad h_{\tau'}(\kappa, \lambda, \mu) < 0$.
2. $\exists \kappa, \lambda, \mu,$ with $(\lambda, \mu) \neq 0$ such that $\langle (a_\tau, b_\tau), (\lambda, -2\mu)\rangle = \kappa$ and $\forall \tau' \neq \tau \quad \langle (a_{\tau'}, b_{\tau'}), (\lambda, -2\mu)\rangle < \kappa$.
3. (a_τ, b_τ) is on the convex hull $\{(a_\tau, b_\tau)\}_{\tau \in \mathcal{T}}$.

To go from (2) to (3) we note that a point is a part of the convex hull if there is a hyperplane (defined by a non-zero vector and a constant) going through this point and such that all other points are strictly on one side of the hyperplane. □

A.3 The proof of Theorem 5

Proof. Starting from equation (4), we get to equation (6) as described in section 2.2 and we recover our functional pruning problem of equation (8) as described in Section 3.1, taking $\mathcal{T} = \{1, \ldots n-1\}$, $a_\tau = \tau$, $b_\tau = \sum_{t=1}^{\tau} x_t$ and $c_\tau = 0$. Thus applying Theorem 4, and then Theorem 1 we get the desired result for $\hat{\tau}(\eta_1)$.

The index maximising the likelihood and its value do depend on η_1, however the set of points we need to build the convex hull does not depend on the value of η_1. Considering $\eta_1 = \arg\max_{\eta_1}\left(\max_{\eta_2} \ell_{\hat{\tau}(.),n}(\eta_1, \eta_2)\right)$ we have $\hat{\tau}(.) = \hat{\tau}(\eta_1)$ and we recover the desired result for $\hat{\tau}(.)$. □

A.4 The proof of Lemma 1

Proof. In the case where all x_τ^d are independent and identically distributed, the property (Ex) holds. To show that property (GP) holds, consider time $n \geq p+2$ and choose any indices $\tau_1, \ldots, \tau_{p+1}$ such that $0 < \tau_1 < \cdots < \tau_{p+1} < n$. We define the following variables for $d = 1, \ldots, p+1$:

$$S_d = \left(\tau_d, \sum_{t=1}^{\tau_d} x_t\right) \in \mathbb{R}^{p+1}\,, \quad M_d = S_d - S_{d-1}\,, \quad \text{with } S_0 = 0\,.$$

It can be observed that $S_d = \sum_{k \leq d} M_k$ for any $d = 1, \ldots, p+1$. Therefore, the linear independence of $S_1, \ldots, S_{p+1}$ is equivalent to the linear independence of $M_1, \ldots, M_{p+1}$. We denote the matrix of size $(p+1) \times (p+1)$ where its d-th column is M_d as M. Consider its transpose matrix M^T. Its first column-vector is filled with integers between 1 and $n-1$. Linear algebra shows that proving linear independence for column vectors in M or in M^T is equivalent. The last p column vectors in M^T are drawn independently with a continuous density and thus are almost surely linearly independent, spanning a hyperplane of dimension p. The probability that a point in $\mathbb{R}^{p+1}$, the first column vector of

M^T, is in this span is 0. In conclusion, for any given n, there are finitely many matrices M to consider, and their probabilities satisfy (GP), ensuring that property (GP) holds. $\square$

B Index Set Equality

The equality $\mathcal{F}_{\mathcal{T}} = \mathcal{G}_{\mathcal{T}}$ can be proved with weaker conditions on A than those of Theorem 2. This equality has a simple geometric interpretation: to be true, any half-line in $Im(A) \times \mathcal{D}_A$ has to intersect the manifold defined by $\{(A(\mu), \mu), \mu \in \mathcal{D}_A\}$.

Theorem 7. *If for all $\tau \in \mathcal{T}$, $c_\tau = 0$ and A is continuous defined on a cone with $Im(A) = \mathbb{R}^+$ such that $A(0) = 0$, $\nabla A(0) = 0$ and $\lim_{\|\mu\|\to\infty} \frac{A(\mu)}{\|\mu\|} = +\infty$ we have $\mathcal{F}_{\mathcal{T}} = \mathcal{G}_{\mathcal{T}}$.*

Proof. Consider τ in $\mathcal{G}_{\mathcal{T}}$. For all $\tau' \neq \tau$, the inverse image of the open set $(0, +\infty)$ by linear functions $g_\tau - g_{\tau'}$ is a non-empty intersection of half-spaces. Moreover, there exists $(\lambda_0, \mu_0) \in Im(A) \times \mathcal{D}_A$ such that for all $\tau' \neq \tau$ in $\mathcal{T}$ we have: $g_\tau(\lambda_0, \mu_0) > g_{\tau'}(\lambda_0, \mu_0)$. Using condition $c_\tau = c_{\tau'} = 0$ and $\alpha > 0$ we get by linearity

$$g_\tau(\alpha\lambda_0, \alpha\mu_0) = \alpha g_\tau(\lambda_0, \mu_0) > \alpha g_{\tau'}(\lambda_0, \mu_0) = g_{\tau'}(\alpha\lambda_0, \alpha\mu_0).$$

Therefore, as all points of the half-line $\{\alpha(\lambda_0, \mu_0), \alpha > 0\}$ belong to this set, it shows that index τ is optimal on an open polyhedral cone.

Suppose that for the considered (λ_0, μ_0), there exists $\alpha_0 > 0$ such that $\lambda_0 = \frac{A(\alpha_0\mu_0)}{\alpha_0}$ then $\alpha_0\mu_0$ is a value in the cone $\mathcal{D}_A$ verifying all the constraints $f_\tau(\alpha_0\mu_0) > f_{\tau'}(\alpha_0\mu_0)$, so that τ also belongs to $\mathcal{F}_{\mathcal{T}}$. In geometric terms, this means that any half-line in the polyhedral cone associated with index τ intersects the manifold defined by $\{(A(\mu), \mu), \mu \in \mathcal{D}_A\}$.

It remains to prove that the proposed assumptions over A are sufficient to find such an alpha proving the set equality. We consider the same half-line $\{\alpha(\lambda_0, \mu_0), \alpha > 0\}$. By linear approximation in 0, we have $A(\epsilon) = A(0) + \langle \epsilon, \nabla A(0) \rangle + o(\|\epsilon\|) = o(\|\epsilon\|)$ and locally, we get $A(\alpha\mu_0) = o(\|\alpha\mu_0\|) < \alpha\lambda_0$ for small α. Notice that as $Im(A) = \mathbb{R}^+$ and the cone is open, we have $\lambda_0 > 0$ and can choose $\|\mu_0\| > 0$. The half-line is therefore above the manifold ("$(\alpha\lambda_0, \alpha\mu_0) > (A(\alpha\mu_0), \alpha\mu_0)$").

Using condition $\lim_{\alpha\to\infty} \frac{A(\alpha\mu_0)}{\|\alpha\mu_0\|} = +\infty$, continuity of A and previous local result $\frac{A(\alpha\mu_0)}{\|\alpha\mu_0\|} = o(1)$ for small α, it shows that there exists α_0 such that $\frac{A(\alpha_0\mu_0)}{\|\alpha_0\mu_0\|} = \frac{\lambda_0}{\|\mu_0\|}$ (as $\frac{\lambda_0}{\|\mu_0\|} > 0$). Thus $(\alpha_0\lambda_0, \alpha_0\mu_0) = (A(\alpha_0\mu_0), \alpha_0\mu_0)$ which leads to $f_\tau(\alpha_0\mu_0) > f_{\tau'}(\alpha_0\mu_0)$ and index τ is in $\mathcal{F}_{\mathcal{T}}$. $\square$

C Stirling numbers of the first kind in terms of the harmonic series

To obtain numerical values of U_n^p and V_n^p of the convex hull of $\{P(\tau)\}_{\tau\in\{1,\dots,n-1\}}$ for $1 \le p \le 5$ we need to know the expressions of $\begin{bmatrix} n \\ m \end{bmatrix}$ for $m = 0, \dots, 6$. As shown in [58], the general formula for the Stirling numbers of the first kind in terms of the harmonic series is

$$\begin{bmatrix} n \\ m \end{bmatrix} = \frac{(n-1)!}{(m-1)!}\omega(n, m-1)\,. \tag{11}$$

The ω-sequence is defined recursively by

$$\omega(n, m) = \mathbb{1}_{m=0} + \sum_{k=0}^{m-1} (1-m)_k \sigma_{k+1} \omega(n, m-1-k)\,,$$

where $\sigma_{k+1} = \sum_{i=1}^{n-1} \frac{1}{i^{k+1}}$ is the partial sums of the harmonic series $\left\{\frac{1}{i^{k+1}}\right\}_{i=1,2,\dots}$ and $(1-m)_k$ are the Pochhammer symbols. Applying the formula (11) for $m = 0, \dots, 6$, we get

$$\begin{bmatrix} n \\ 0 \end{bmatrix} = 1\,, \quad \begin{bmatrix} n \\ 1 \end{bmatrix} = (n-1)!\,, \begin{bmatrix} n \\ 2 \end{bmatrix} = (n-1)!\sigma_1\,,$$

$$\begin{bmatrix} n \\ 3 \end{bmatrix} = \frac{(n-1)!}{2}(\sigma_1^2 - \sigma_2)\,, \quad \begin{bmatrix} n \\ 4 \end{bmatrix} = (n-1)!\left(\frac{\sigma_1^3}{6} - \frac{\sigma_1\sigma_2}{2} + \frac{\sigma_3}{3}\right)\,,$$

$$\begin{bmatrix} n \\ 5 \end{bmatrix} = (n-1)! \left(\frac{\sigma_1^4}{24} - \frac{\sigma_1^2\sigma_2}{4} + \frac{\sigma_1\sigma_3}{3} + \frac{\sigma_2^2}{8} - \frac{\sigma_4}{4} \right),$$
$$\begin{bmatrix} n \\ 6 \end{bmatrix} = (n-1)! \left(\frac{\sigma_1^5}{120} - \frac{\sigma_1^3\sigma_2}{12} + \frac{\sigma_1^2\sigma_3}{6} + \frac{\sigma_1\sigma_2^2}{8} - \frac{\sigma_1\sigma_4}{4} - \frac{\sigma_2\sigma_3}{6} + \frac{\sigma_5}{5} \right).$$

D Detection of multiple changes in the parameter of a distribution in the exponential family

We define the set $\mathcal{M}_n^K$ of all possible segmentations in K changepoints of time series $\{y_t\}_{t=1,\dots,n}$ as:

$$\mathcal{M}_n^K = \{\tau = (\tau_0, \tau_1, \dots, \tau_K, \tau_{K+1}) \in \mathbb{N}^{K+2} | 0 = \tau_0 < \tau_1 < \dots < \tau_K < \tau_{K+1} = n\}.$$

For any segmentation τ in M_n^K we define its size as $|\tau| = K$. We denote $\mathcal{M}_n^\infty$ as the set of all possible segmentations of $\{y_t\}_{t=1,\dots,n}$:

$$\mathcal{M}_n^\infty = \bigcup_K M_n^K , \tag{12}$$

and take the convention that $M_n^{\infty-1} = M_n^\infty$.

From a statistical perspective given a subset of all segmentations $\mathcal{M}_n^K$ for K in $\mathbb{N} \cup \{\infty\}$ it makes sense to recover the segmentation maximising a penalised likelihood of our (transformed as in Section 2.2) data $\{x_t\}_{t=1,\dots,n}$:

$$C_n^K = \max_{\tau \in \mathcal{M}_n^K} \left\{ \sum_{k=1}^{|\tau|+1} \max_{\eta_k} \left[\sum_{t=\tau_{k-1}+1}^{\tau_k} 2\log(f_X(x_t|\eta_k)) - \beta \right] \right\} , \tag{13}$$

where β is a penalty value in $\mathbb{R}^+$.

The previous equation, encompasses two common multiple changepoints strategies. In the first, that we call Segment Neighbourhood (SN) [22] the user is given a maximum number of changepoints K_{max} and solve equation (13) for all $K \leq K_{max}$ (setting β to 0). In a second step all these solutions are compared using an appropriate model selection criteria such as [28, 69] to select the number of changepoints. In the second strategy, that we call Optimal Partitioning (OP) [25] the position and number of changepoints are estimated in one go solving (13) for $K = \infty$ and picking an appropriate value for β such as $2\hat{\sigma}^2 \log(n)$ in a Gaussian setting with $p = 1$ [70] where $\hat{\sigma}^2$ is an estimate of the Gaussian variance.

Finding the maximum of (13) in either case can be done using dynamic programming. For any segmentation τ with $|\tau| > 1$, observe that the first $|\tau|$ terms in the previous sum are by definition smaller than $C_{\tau_{|\tau|}}^{K-1}$. Hence, we only need to consider the position of the last changepoint and we recover the update rule of the SN [22] and OP [25] algorithms:

$$C_t^K = \max_{i<t} \left\{ C_i^{K-1} + \max_{\eta} \left[\sum_{s=i+1}^{t} 2\log(f_X(x_s|\eta)) - \beta \right] \right\} . \tag{14}$$

With this recursion solved for increasing t, and, in the case of SN, increasing K.

Directly solving this recursion has a computational cost that is quadratic in n. This has led to pruning ideas that speed up the computation whilst still solving the maximisation problem exactly. The idea of these pruning rules is that at any time t we need only consider a subset of possible changepoint locations. The most efficient pruning rules are based on so-called functional pruning (as in the pDPA or FPOP algorithms [32, 33]). The idea is to consider the best segmentation of the data to time t as a function of the parameter of the final segment. For each value of this parameter there will be an optimal position of the most recent changepoint prior to t, and at the next iteration we need only consider these positions and t for the possible position of the most recent changepoint prior to $t + 1$. Implementing this algorithm requires calculating regions of the parameter space which are optimal for each value of the most recent changepoint. When $p = 1$ this is straightforward as we are manipulating intervals, but for $p > 1$ this becomes challenging, and often computationally prohibitive, as it involves computing the intersection and subtraction of many convex sets in $\mathbb{R}^p$ [35, 36].

D.1 Connection between the Detection of Multiple changepoints and the half-space intersection problem

In Section 4, we established that the maximum likelihood estimator of a single changepoint problem corresponds to a vertex on the hull of a particular set of points.This result is primarily based on Theorem 4, which follows from the Half-space Intersection Theorem 3.

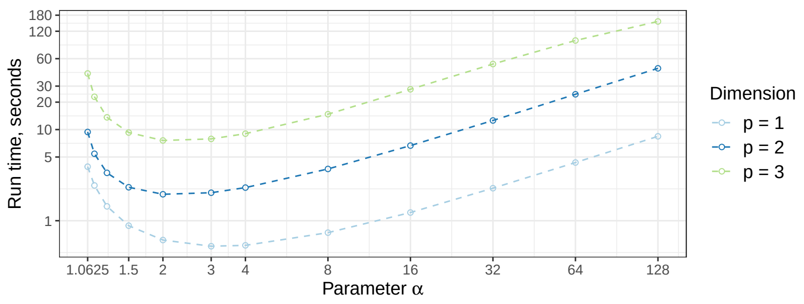


Figure 10: Run time of MdFOCuS0 as a function of α for p-variate time series with $n = 10^5$ data points in dimension $p = 1, 2$ and 3. We simulated 100 i.i.d. Gaussian data $\mathcal{N}_p(0, I_p)$ and report the average run time.

In the following theorem, we establish that the last changepoint of the maximum likelihood segmentation (see equation (13)) of a multiple changepoint problem corresponds to a supporting hyperplane of a particular half-space intersection problem defined as in Theorem 3.

Theorem 8. *The last changepoint $\hat{\imath}$ of the segmentation $\hat{\tau}$ maximising the likelihood* (13) *is such that $H_{\hat{\imath}}$ is a supporting hyperplane of $\cap_i H_i$ where i is in $\{1, \dots, n-1\}$ and H_i is defined as the hyperplane*

$$H_i = \{\kappa, \lambda, \mu \mid h_i(\kappa, \lambda, \mu) = a_i\lambda - 2\langle b_i, \mu\rangle + c_i - \kappa \leq 0\},$$

with $a_i = i$, $b_i = \sum_{t=1}^{i} x_t$ and $c_i = C_i^{K-1} - \sum_{t=1}^{i} B'(x_t)$.

Proof. We start from equation (13). There is a value $\hat{\eta}$ for which $\hat{\tau}$ is optimal. Now if we subtract $\sum_{t=1}^{n}\left\{2\log(f_X(x_t|\eta)) - \beta\right\}$, $\hat{\eta}$ is still optimal for $\eta = \hat{\eta}$. Now applying Theorem 3 and restricting our attention to the last changepoint as explain for equation (7) we get the desired result. $\square$

Contrary to the single changepoint case, we do not propose an algorithm based on this connection. Nonetheless, in a manner similar to the MdFOCuS algorithm, we think one could restrict the set of last changepoints to consider in the standard SN or OP algorithm using the output of an half-space intersection algorithm.

E Additional Empirical evaluation of the MdFOCuS algorithm

E.1 Run time as a function of a slope parameter

Now we consider the empirical run times of the MdFOCuS algorithm with a known pre-change parameter η_1 as a function of a slope parameter α. Using the implementation of Algorithm 1 for a change in the mean of multi-dimensional Gaussian signal (see Table 1) in R/C++ using the qhull library (see link in the Supplementary Material) we studied how the α parameter affects its running time. To do this, we generate 100 time series with $n = 10^5$ data points (without changepoints, and $x_t \sim \mathcal{N}_p(0, I_p)$ i.i.d) in dimension $p = 1, 2$ and 3. In Figure 10 we report the average run time as a function of α for MdFOCuS0 (that is for a known value of η_1) and values of α in $\left\{1 + \{2^i\}_{i\in\{-4,\dots,0\}}, 3, \{2^j\}_{j\in\{2,\dots,7\}}\right\}$. The value of $maxSize$ was always initialised to $p + 2$ and β always set to 1. We observe a minimum run time for α in $[2, 4]$. As a consequence in the rest of the simulations we always set α to 2.

E.2 Numerical study on 5-dimensional sequences

We replicate the simulation study of Section 4.3.2 for 5-dimensional time series. The results are presented in Tables 6-9.

E.3 Multidimensional Poisson Model

We simulated time series with a Poisson distribution : $x_{1:n}$ ($x_t \sim \mathcal{P}(\mathbf{1}_p), t = 1, \dots, n$). We varied p and n as in Section 4.2.

Table 6: Average detection delay in the pre-change mean known scenario with 5-dimensional time series for a change at time $t = 500$. Best results per row are highlighted in bold.

magnitude	density	FOCuS0	MdFOCuS0	ocd (ora)
0.25	1	265.62	319.50	**241.83**
0.25	2	286.83	307.84	**280.51**
0.25	3	**301.59**	304.99	307.18
0.25	4	262.60	**244.98**	271.74
0.25	5	314.72	**295.74**	332.29
0.50	1	75.81	86.82	**68.02**
0.50	2	83.73	86.70	**75.64**
0.50	3	91.83	87.44	**83.18**
0.50	4	78.70	**67.85**	73.73
0.50	5	95.41	**83.90**	88.23
1.00	1	**20.57**	23.49	21.99
1.00	2	23.01	**22.99**	23.59
1.00	3	25.02	**22.93**	25.02
1.00	4	22.48	**18.65**	20.99
1.00	5	26.83	**22.78**	25.70

Table 7: Standard deviation of the detection delay in the pre-change mean known scenario with 5-dimensional time series for a change at time $t = 500$. Best results per row are highlighted in bold.

magnitude δ	FOCuS0	MdFOCuS0	ocd(oracle)
0.25	10.28	**8.85**	23.61
0.50	4.00	**2.73**	4.20
1.00	0.92	**0.40**	0.99

Table 8: Average detection delay in the pre-change mean unknown scenario with 5-dimensional time series for a change at time $t = 500$. Best results per row are highlighted in bold.

magnitude δ	density	FOCuS	FOCuS0 (est)	MdFOCuS	ocd (est)
0.50	1	**89.30**	108.70	105.47	95.37
0.50	2	104.86	125.42	105.22	**102.48**
0.50	3	114.19	141.38	**102.65**	110.18
0.50	4	97.32	130.23	**79.75**	88.44
0.50	5	123.96	158.55	**104.24**	106.33
1.00	1	**21.69**	28.22	24.12	28.12
1.00	2	24.59	33.16	**24.02**	28.94
1.00	3	27.26	37.23	**24.09**	29.51
1.00	4	24.59	35.73	**19.83**	23.72
1.00	5	29.76	42.84	**24.31**	29.11
2.00	1	**5.95**	7.67	6.63	7.96
2.00	2	6.87	9.30	**6.72**	8.07
2.00	3	7.40	10.31	**6.67**	8.00
2.00	4	6.93	9.80	**5.60**	6.57
2.00	5	8.31	11.82	**6.72**	8.05

Table 9: Standard deviation of the detection delay in the pre-change mean unknown scenario with 5-dimensional time series for a change at time $t = 500$. Best results per row are highlighted in bold.

magnitude δ	FOCuS	FOCuS0 (est)	MdFOCuS	ocd (est))
0.50	8.44	8.26	**7.02**	4.32
1.00	1.23	2.04	**0.32**	0.85
2.00	0.28	0.50	**0.14**	0.18

E.3.1 Empirical validation of the bound

Using the same procedure as in Section 4.2, we evaluate the number of faces and vertices of the convex hull $\{P(\tau)\}_{\tau\in\{1\dots,n-1\}}$. The results are presented in Figure 11. We note that the results are similar to the Gaussian case (see Figure 3).

E.3.2 Empirical run times of the algorithm as a function of dimension

Using the same procedure as in Section 4.3.1 we evaluate the run times of MdFOCuS0 and MdFOCuS for Poisson distribution. The results are presented in Figure 12. We note that the results are similar to the Gaussian case (see Figure 4).

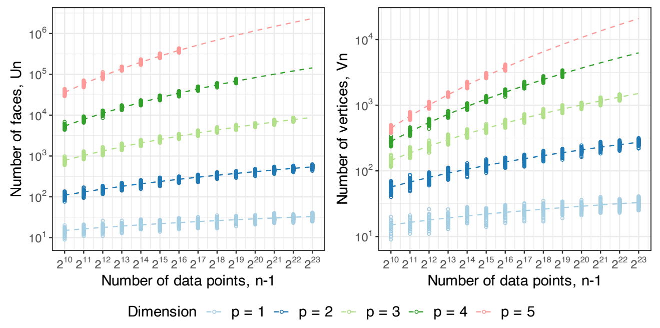


Figure 11: The number of faces (left) and vertices (right) of the convex hull built from the set of points $\{P(i)\}_{i\in\{1\dots,n-1\}}$ for dimension $1 \le p \le 5$. We simulated 100 i.i.d. p-variate Poisson data $\mathcal{P}(\mathbf{1}_p)$ for each n from $(2^{11}+1)$ to $(2^{23}+1)$. Dashed lines correspond to the expected number of faces (left) and vertices (right) presented in Theorem 6. We consider a maximum running time of 10 minutes for the *Quickhull* algorithm. That is why we don't have data points for $p = 3, 4$ and 5 for large signals.

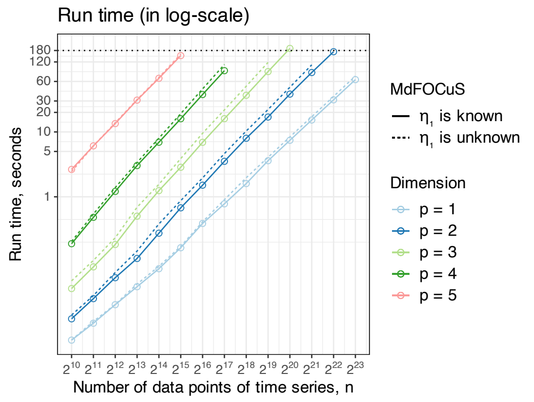


Figure 12: Run times in seconds of MdFOCuS with known (dashed line) and unknown(full line) pre-change parameter η_1, both with $\alpha = 2$, $\beta = 1$ and $maxSize = 14$, in dimension $p = 1, \dots, 5$ using time series $x_{1:n}$ (without change, $x_t \sim \mathcal{P}(\mathbf{1}_p)$ i.i.d). Run times are averaged over 100 data sets. We considered a maximum running time of 3 minutes (horizontal dotted black line) for the algorithms. That is why we do not report run times for $p = 3, 4$ and 5 for large signals. We report here the slope of the curve obtained using a simple linear regression: for known η_1: ≈ 1.05 ($p = 1$), ≈ 1.16 ($p = 2$), ≈ 1.24 ($p = 3$), ≈ 1.26 ($p = 4$) and ≈ 1.16 ($p = 5$); for unknown η_1: ≈ 1.06 ($p = 1$), ≈ 1.18 ($p = 2$), ≈ 1.26 ($p = 3$), ≈ 1.29 ($p = 4$) and ≈ 1.19 ($p = 5$).

F Dyadic MdFOCuS : an algorithm with a quasi-linear expected complexity

In this appendix we present an algorithm for online changepoint detection. In essence, as in the MdFOCuS algorithm of the main text, it maintains a set of points that is slightly larger than the set of points on the hull. It does so applying the *QuickHull* (or any convex hull) algorithm on successive and non-overlapping deterministic chunks of the points. This deterministic choice of chunks is likely sub-optimal but it allows for a precise quantification of the expected complexity of the algorithm under the assumption of Theorem 6 (or Corollary 1) and under the assumption that the convex hull algorithm is at worst quadratic. To be specific we obtain a $\mathcal{O}(n\log(n)^{p+1})$ bound on the expected time complexity.

The algorithm we propose (as the one of the main text) considers at least all points on the hull to maximise the likelihood for every new data point. For a fixed p following, Theorem 6 and its corollary the number of points on the hull at step i is in $\Omega((\log(i))^p)$. To recover a lower bound on the complexity, we first sum overall $i \le n$, and then consider only $i \ge n/2$ to recover that the overall number of points we will consider is in $\Omega(n/2(\log(n/2))^p)$ and thus $\Omega(n(\log(n))^p)$. Relative to that our algorithm loose just one $\log(n)$ term, we thus argue our bound is relatively tight.

The assumption on the quadratic complexity is true for *QuickHull* in dimension 2 and 3 (so for $p \leq 2$) [48, 63] and using the algorithm of Chazelle [64] it is also true in dimension 4 (so for $p \leq 3$). Still following Chazelle this is no longer true for any dimension strictly larger than 4 and therefore $p > 3$.

We begin by two preliminary results. The first is a crude bound on the expectation of the square of the number of points on the convex hull of n points (that is $(V_n^p)^2$ following the notation of Section 4.2) under the assumptions of Theorem 6. The second result bounds the expected complexity of computing the hull of $2n$ points if we already know the points on the hull of the first n points and the last n points. In all that follows we will assume the two following assumptions.

Assumption 1. *In any dimension $p \geq 1$, there are positive constants c_1 and c_2 for which the expected number of vertices, $\mathbb{E}[V_n^p]$, can be bounded as follows:*

$$\mathbb{E}[V_n^p] \leq c_1 \log(n)^p + c_2.$$

This is true under the conditions of Corollary 1.

Assumption 2. *We assume that the convex hull algorithm we are using is at worst quadratic in time and assume there exist two positive constants c_3 and c_4 such that for n points the time complexity is $Time(n) \leq c_3 n^2 + c_4$.*

We now state our bound on the expectation of the square of the number of points on the hull V_n^p.

Lemma 2.

$$\mathbb{E}[(V_n^p)^2] \leq c_1 n \log(n)^p + c_2 n. \tag{15}$$

Proof. V_n^p is bounded by 0 and $n-1$. It follows that

$$\mathbb{E}[(V_n^p)^2] = \sum_{i=0}^{n} \mathbb{P}(V_n^p = i) i^2 \leq n \sum_{i=0}^{n} \mathbb{P}(V_n^p = i) i \leq n \mathbb{E}[V_n^p] \leq c_1 n \log(n)^p + c_2 n. \tag{16}$$

We get the last inequality with Assumption 1. □

We now state our bound on the complexity of merging the hull of two successive chunks.

Lemma 3. *Consider we have $2n$ data points. Assume we know $\mathcal{W}_1$ the index of the points on the hull of $\{P(\tau)\}_{1 \leq \tau \leq n}$ and $\mathcal{W}_2$ the index of the points on the hull of $\{P(\tau)\}_{n+1 \leq \tau \leq 2n}$. Under the condition of Corollary 1 and Assumption 2 computing the index of the points on the hull of $\{P(\tau)\}_{1 \leq \tau \leq 2n}$ can be done in less than $4c_1c_3 n(\log n)^p + 4c_2c_3 n + c_4$ time.*

Proof. We will compute those points as the points on the hull of $\{P(\tau)\}_{\tau \in \mathcal{W}_1 \cup \mathcal{W}_2}$. By Assumption 2 the time complexity of processing these $|\mathcal{W}_1| + |\mathcal{W}_2|$ points is

$$Time\left(|\mathcal{W}_1| + |\mathcal{W}_2|\right) \leq c_3 \left(|\mathcal{W}_1| + |\mathcal{W}_2|\right)^2 + c_4 \leq 2c_3\left(|\mathcal{W}_1|^2 + |\mathcal{W}_2|^2\right) + c_4.$$

Now taking the expectation and noting that under the assumption of Corollary 1 the distribution of $|\mathcal{W}_1|$ is the same as the distribution of $|\mathcal{W}_2|$ we get

$$\mathbb{E}\left(Time\left(|\mathcal{W}_1| + |\mathcal{W}_2|\right)\right) \leq 4c_3 \mathbb{E}\left(|\mathcal{W}_1|^2\right) + c_4.$$

Using (15) we get that the expected complexity is

$$\mathbb{E}\left(Time\left(|\mathcal{W}_1| + |\mathcal{W}_2|\right)\right) \leq 4c_1c_3 n(\log n)^p + 4c_2c_3 n + c_4. \tag{17}$$

□

F.1 Dyadic algorithm for MdFOCuS

F.1.1 Set of candidates

Recall that at time n we would ideally only store the vertices on the hull of $\{P(\tau)\}_{\tau<n}$. In the following algorithm we store the vertices on the hull of chunks of the form $\{k2^q+1, \ldots, (k+1)2^q\}$ for all $q \geq q_{min}$. To be specific, we consider the base 2 description of $n : n = \sum_{q=0}^{\lfloor \log_2(n) \rfloor} \overline{n}_q 2^q$, with $\overline{n}_q = 0$ or 1, and define for q in $\{0, \ldots, \lfloor \log_2(n-1) \rfloor\}$ the subset $\mathcal{U}_n^q$ of $\{1, \ldots, n-1\}$ as

$$\mathcal{U}_n^q = \{\tau | \sum_{q'=q+1}^{\lfloor \log_2(n-1) \rfloor} \overline{(n-1)}_{q'} 2^{q'} < \tau \leq \sum_{q'=q}^{\lfloor \log_2(n-1) \rfloor} \overline{(n-1)}_{q'} 2^{q'} \},$$

Table 10: A few $\mathcal{U}_n^q$ sets.

n	$q=4$	$q=3$	$q=2$	$q=1$	$q=0$
31	$\{1,\dots 16\}$	$\{17,\dots 24\}$	$\{25,\dots 28\}$	$\{29,30\}$	$\emptyset$
30	$\{1,\dots 16\}$	$\{17,\dots 24\}$	$\{25,\dots 28\}$	$\emptyset$	$\{29\}$
29	$\{1,\dots 16\}$	$\{17,\dots 24\}$	$\{25,\dots 28\}$	$\emptyset$	$\emptyset$
28	$\{1,\dots 16\}$	$\{17,\dots 24\}$	$\emptyset$	$\{25,26\}$	$\{27\}$
27	$\{1,\dots 16\}$	$\{17,\dots 24\}$	$\emptyset$	$\{25,26\}$	$\emptyset$
26	$\{1,\dots 16\}$	$\{17,\dots 24\}$	$\emptyset$	$\emptyset$	$\{25\}$
25	$\{1,\dots 16\}$	$\{17,\dots 24\}$	$\emptyset$	$\emptyset$	$\emptyset$
24	$\{1,\dots 16\}$	$\emptyset$	$\{17,\dots 20\}$	$\{21,22\}$	$\{23\}$
23	$\{1,\dots 16\}$	$\emptyset$	$\{17,\dots 20\}$	$\{21,22\}$	$\emptyset$
22	$\{1,\dots 16\}$	$\emptyset$	$\{17,\dots 20\}$	$\emptyset$	$\{21\}$
21	$\{1,\dots 16\}$	$\emptyset$	$\{17,\dots 20\}$	$\emptyset$	$\emptyset$
20	$\{1,\dots 16\}$	$\emptyset$	$\emptyset$	$\{17,18\}$	$\{19\}$

with the convention that for $q = \lfloor \log_2(n-1) \rfloor$ the sum $\sum_{q'=q+1}^{\lfloor \log_2(n-1)\rfloor} \overline{(n-1)}_{q'} 2^{q'}$ is equal to 0.

We provide a few examples of these sets for $n = 20$ to $n = 31$ in Table 10. It can be observed that often $\mathcal{U}_n^q$ differs from $\mathcal{U}_{n+1}^q$ only for small values of q.

Now note that for any $q_{min} \leq \lfloor \log_2(n-1) \rfloor$

$$\bigcup_{q=q_{min}}^{\lfloor \log_2(n-1)\rfloor} \mathcal{U}_n^q \quad \cup \quad \{\tau | \sum_{q'=q_{min}}^{\lfloor \log_2(n-1)\rfloor} \overline{(n-1)}_{q'} 2^{q'} < \tau \leq n-1\},$$

is a partition of $\{1, \dots, n-1\}$. The right-side set of this partition has at most $2^{q_{min}}$ elements. We use this partition to update the set of candidates. To be specific our algorithm maintains

$$\mathcal{T}_n = \bigcup_{q=q_{min}}^{\lfloor \log_2(n-1)\rfloor} \mathcal{W}_n^q \quad \cup \quad \left(\{P(\tau) | \sum_{q'=q_{min}}^{\lfloor \log_2(n-1)\rfloor} \overline{(n-1)}_{q'} 2^{q'} < \tau \leq n-1\} \right),$$

with $\mathcal{W}_n^q = \textsc{Hull}\left(\{P(\tau)\}_{\tau \in \mathcal{U}_n^q}\right)$. Note that $\mathcal{T}_n$ contains all the points (or index of the points) on the hull of $\{P(\tau)\}_{\tau \in \{1,\dots,n-1\}}$.

In the next paragraphs we bound the expected number of points in $\mathcal{T}_n$ and then explain how we compute $\mathcal{T}_{n+1}$ from $\mathcal{T}_n$.

Lemma 4. *For a fixed q_{min}, under Assumption 1 we have*

$$\mathbb{E}(|\mathcal{T}_n|) \leq 2c_1(\log n)^{p+1} + 2c_2 \log n + 2^{q_{min}}. \tag{18}$$

Proof. For a fixed q_{min} using Assumption 1 for all $\mathcal{U}_n^q$ we get that

$$\begin{aligned}
\mathbb{E}(|\mathcal{T}_n|) &\leq \sum_{q=q_{min}}^{\lfloor \log_2(n-1)\rfloor} (c_1(\log 2^q)^p + c_2) + 2^{q_{min}} \\
&\leq \sum_{q=q_{min}}^{\lfloor \log_2(n-1)\rfloor} (c_1(\log n)^p + c_2) + 2^{q_{min}} \\
&\leq 2c_1(\log n)^{p+1} + 2c_2 \log n + 2^{q_{min}}.
\end{aligned}$$

We get the last inequality using the fact that we sum over at most $\log_2(n)$ terms and $1/\log(2) \leq 2$. □

F.1.2 Updating the set of candidates

To compute $\mathcal{T}_{n+1}$ from scratch we need to apply the hull algorithm on all $\mathcal{U}_{n+1}^q$ for $q \geq q_{min}$. Assuming we already computed $\mathcal{T}_n$ computing $\mathcal{T}_{n+1}$ is a bit easier. This is because it is often the case that $\mathcal{U}_{n+1}^q = \mathcal{U}_n^q$. Therefore we directly get the indices of the point on the hull of $\mathcal{U}_{n+1}^q$ as $\mathcal{U}_n^q \cap \mathcal{T}_n$ because

$$\mathcal{W}_n^q = \textsc{Hull}\left(\{P(\tau)\}_{\tau \in \mathcal{U}_n^q}\right) = \mathcal{U}_n^q \cap \mathcal{T}_n \,.$$

Algorithm 2 formalise this idea and iteratively merge chunks of size 2^q to take advantage of Lemma 3.

Algorithm 2 Update of the set of candidates

```
1:  Input : $n$ and $\mathcal{T} = \mathcal{T}_n$
2:  $\mathcal{T} \leftarrow \mathcal{T} \cup \{n\}$
3:  $q \leftarrow q_{min}$
4:  while $remainder(n/2^q) = 0$ do
5:      $\mathcal{T}' \leftarrow \{\tau \in \mathcal{T} | \tau > n - 2^q\}$
6:      $\mathcal{T}'' \leftarrow$ QUICKHULL($\{P(\tau)\}_{\tau \in \mathcal{T}'}$)
7:      $\mathcal{T} \leftarrow \mathcal{T} \setminus (\mathcal{T}' \setminus \mathcal{T}'')$          ▷ discard vertices in $\mathcal{T}'$ not in $\mathcal{T}''$
8:      $q \leftarrow q + 1$
9:  end while
10: return $\mathcal{T} = \mathcal{T}_{n+1}$
```

Remark 7. *In Algorithm 2 we assume that if the set of points $\{P(\tau)\}_{\tau \in \mathcal{T}'}$ doesn't define a proper volume in dimension $p+1$ then the output of* QUICKHULL$(\{P(\tau)\}_{\tau \in \mathcal{T}'})$ *is simply $\mathcal{T}'$. If we choose q_{min} such that $2^{q_{min}}$ is larger than $p+2$ this is unlikely to happen for data drawn from distribution with a continuous density.*

We now continue with an informal description of Algorithm 2. Looking at line 4 when we include a new data point $n+1$ two cases can happen.

- n is not a multiple of $2^{q_{min}}$. In that case for all $q \geq q_{min}$ we have $\mathcal{U}^q_{n+1} = \mathcal{U}^q_n$. We just need to add n as a possible changepoint to $\mathcal{T}_n$ to recover $\mathcal{T}_{n+1}$. This was done at line 2.
- n is a multiple of $2^{q_{min}}$. In that case for some $q \geq q_{min}$ we have that $\mathcal{U}^q_{n+1}$ is different from $\mathcal{U}^q_n$. We therefore need to update the set of points using the hull algorithm.

For the later case let us define $q_n \geq q_{min}$ as the largest integer such that 2^{q_n} divides n. Before we explain the update of line 5, 6 and 7 note the following facts.

- In base 2, n ends with a one followed by q_n zeros. In other words $\overline{n}_{q_n} = 1$ and for all $q < q_n$ $\overline{n}_q = 0$. Thus all $\mathcal{U}^q_{n+1}$ with $q < q_n$ should be empty and $\mathcal{U}^{q_n}_{n+1}$ is not.
- In base 2, $n-1$ ends with a zero followed by q_n ones. In other words $\overline{(n-1)}_{q_n} = 0$ and for all $q < q_n$ $\overline{(n-1)}_q = 1$. Thus all $\mathcal{U}^q_n$ with $q < q_n$ are not empty.
- We know the index of the points on the hull for all set of indices $\mathcal{U}^q_n$ with $q < q_n$:
$$\mathcal{W}^q_n = \text{HULL}\left(\{P(\tau)\}_{\tau \in \mathcal{U}^q_n}\right) = \mathcal{U}^q_n \cap \mathcal{T}_n.$$
- Finally, $\mathcal{U}^{q_n}_{n+1} = (\cup_{q_{min} \leq q < q_n} \mathcal{U}^q_n) \cup \{\tau | n - 2^{q_{min}} < \tau \leq n\}$.

Our goal is to get the point (or indices of these points) on the hull of $P(\tau)$ for τ in $\mathcal{U}^{q_n}_{n+1}$. To exploit the merging bound of (17) Algorithm 2 iteratively apply the hull algorithm on larger and larger chunks of the data trimming the points that are not on the hull. In the while loop, from line 4 to 9, of Algorithm 2 we can distinguish two cases.

1. For $q = q_{min}$ on line 5 and 6 the algorithm applies the hull algorithm on the 2^q right-most points: HULL$(\{P(\tau)\}_{n-2^q < \tau \leq n})$. It then discards on line 7 all indices larger than $n - 2^q$ that are not on this hull.
2. If $q_{min} < q \leq q_n$ on line 5 and 6 the algorithm applies the hull algorithm on indices that are larger than $n - 2^q$. In essence it is applying the hull algorithm on the points computed during the previous while step (indices larger than $n - 2^{q-1}$ and in $\mathcal{T}$) with those of $\mathcal{W}^q_n = \text{HULL}\left(\{P(\tau)\}_{\tau \in \mathcal{U}^q_n}\right) = \mathcal{U}^q_n \cap \mathcal{T}_n$ (indices between $n - 2^q$ and $n - 2^{q-1}$, and in $\mathcal{T}$). A bit more formally, calling $\mathcal{W}'^q_{n+1}$ the ouput of line 6 ($\mathcal{T}''$) at step q of the while loop, we are merging $\mathcal{W}^q_n$ and $\mathcal{W}'^{q-1}_{n+1}$ as follows:
$$\mathcal{W}'^q_{n+1} = \text{HULL}\left(\{P(\tau)\}_{\tau \in \mathcal{W}^q_n \cup \mathcal{W}'^{q-1}_{n+1}}\right).$$
On line 7 the algorithm is discarding all indices larger than $n - 2^q$ that are not on this hull $\mathcal{W}'^q_{n+1}$.

Proceeding in such a way at each step we merge the points on the hull of two successive chunks of size 2^{q-1} to recover the point on the hull of a chunk of size 2^q. Figure 13 is a schematic representation of these successive fusions (applications of the hull algorithm).

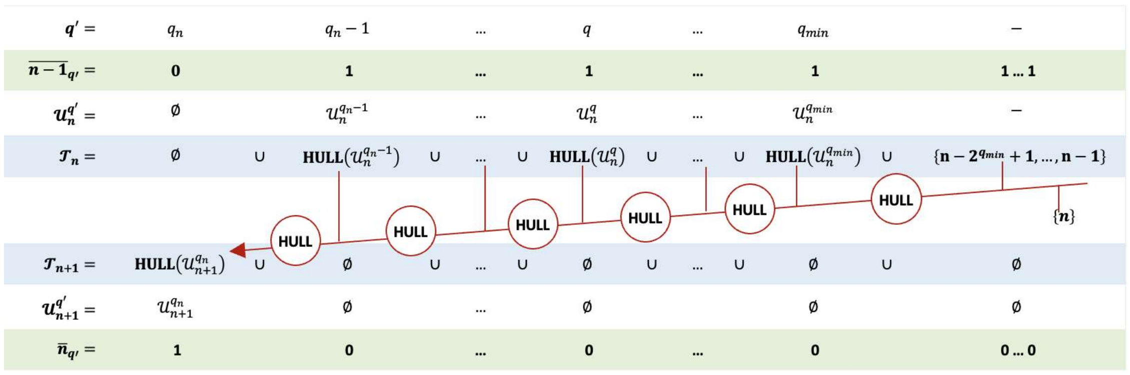

$q' =$ q_n $q_n - 1$ ... q ... q_{min} –
$\overline{n-1}_{q'} =$ 0 1 ... 1 ... 1 1 ... 1
$\mathcal{U}_n^{q'} =$ $\emptyset$ $\mathcal{U}_n^{q_n-1}$ ... $\mathcal{U}_n^q$ ... $\mathcal{U}_n^{q_{min}}$ –
$\mathcal{T}_n =$ $\emptyset$ $\cup$ HULL$(\mathcal{U}_n^{q_n-1})$ $\cup$ ... $\cup$ HULL$(\mathcal{U}_n^q)$ $\cup$ ... $\cup$ HULL$(\mathcal{U}_n^{q_{min}})$ $\cup$ $\{n - 2^{q_{min}} + 1, \ldots, n-1\}$
HULL HULL HULL HULL HULL HULL
$\{n\}$
$\mathcal{T}_{n+1} =$ HULL$(\mathcal{U}_{n+1}^{q_n})$ $\cup$ $\emptyset$ $\cup$ ... $\cup$ $\emptyset$ $\cup$ ... $\cup$ $\emptyset$ $\cup$ $\emptyset$
$\mathcal{U}_{n+1}^{q'} =$ $\mathcal{U}_{n+1}^{q_n}$ $\emptyset$ ... $\emptyset$ $\emptyset$ $\emptyset$
$\overline{n}_{q'} =$ 1 0 ... 0 ... 0 0 ... 0

Figure 13: Schematic representation of the while loop of Algorithm 2 for q_n being the largest q such that 2^q divides n. First applying the hull algorithm on $\{n - 2^{q_{min}} + 1, \ldots, n\}$ (on the right) following the red arrow we iteratively aggregate the result with the hull of points in $\mathcal{U}_n^q$ to recover the point on the hull of $\mathcal{U}_{n+1}^{q_n}$ (on the left). In such a way we effectively retrieve $\mathcal{T}_{n+1}$ as all $\mathcal{U}_{n+1}^q$ are empty for $q < q_n$.

Example 1. *In this paragraph we explain how the algorithm proceeds for $n = 24$ and $q_{min} = 2$. In base 2, $n - 1 = 10111$. Therefore (see the definition of set $\mathcal{T}_n$) we already computed the points on the hull of $P(\tau)$ for τ in $\mathcal{U}_{24}^4 = \{1, \ldots 16\}$ and τ in $\mathcal{U}_{24}^2 = \{17, \ldots, 20\}$, and additionally we store all indices τ larger than $24 - 2^2$: $\{21, 22, 23\}$. When we include a new observation $n + 1 = 25$, we need to include a possible change at $n = 24$ and here is what we do with the while loop (starting with $q = 2$):*

- *for $q = 2$: we run Quickhull on points with indices strictly larger than 20 : $\{21, 22, 23, 24\}$;*
- *for $q = 3$: to recover $\mathcal{U}_{25}^3$ we run Quickhull on points with indices strictly larger than 16 that is the points obtain in the previous step ($q = 2$) union those of the hull of $\mathcal{U}_{24}^2$ (that were computed for $n + 1 = 21$ as the point on the hull of $\mathcal{U}_{21}^2$).*
- *for $q = 4$: we stop as the remainder of $24/2^4$ is not 0.*

We now give the time expected time complexity of the algorithm.

Lemma 5. *Under the assumption of Corollary 1 and Assumption 2, the expected time complexity of Algorithm 2 is*

$$\mathbb{E}\left(Time(Algorithm\ 2)\right) = \mathcal{O}\left(n \log(n)^{p+1}\right). \tag{19}$$

Proof. Considering all time steps up to $n + 1$ the algorithm used the *QuickHull* algorithm on all successive (and non overlapping) chunks of size 2^q for q larger than q_{min} and smaller than $\lfloor \log_2(n) \rfloor$.

For the smallest scale, $n' = 2^{q_{min}}$, we have $n/2^{q_{min}}$ chunks and run *QuickHull* on all points. Using Assumption 2, the worst time per chunk is less than $c_3(2^{q_{min}})^2 + c_4$, and summing overall chunks we get $nc_3 2^{q_{min}} + \frac{nc_4}{2^{q_{min}}}$ operations. So the smallest scale is in $\mathcal{O}(n)$.

We will now successively consider all larger scales $n' = 2^q$ for $q_{min} < q \leq \lfloor \log_2(n) \rfloor$. Note that the hull of a chunk of size $n' = 2^q$ is obtained combining the hulls of two smaller chunks of size 2^{q-1}. For any such chunk we can thus exploit Lemma 3 and equation (17) with $|\mathcal{W}_1| = |\mathcal{W}_2| = n'/2 = 2^{q-1}$ and we get

$$\mathbb{E}\left(Time(|\mathcal{W}_1| + |\mathcal{W}_2|)\right) \leq 4c_1c_3 2^{q-1}(\log 2^{q-1})^p + 4c_2c_3 2^{q-1} + c_4\,.$$

Multiplying by $n/2^q$ to consider all chunks of size 2^q we get that the time to process scale 2^q is

$$\begin{aligned}
\mathbb{E}\left(Time(\text{“Scale } 2^q\text{”})\right) &\leq \frac{n}{2^q}\left(4c_1c_3 2^{q-1}(\log 2^{q-1})^p + 4c_2c_3 2^{q-1} + c_4\right) \\
&\leq 4c_1c_3\frac{n}{2}(\log 2^{q-1})^p + 4c_2c_3\frac{n}{2} + c_4\frac{n}{2^q} \\
&\leq 4c_1c_3\frac{n}{2}(\log n)^p + 4c_2c_3\frac{n}{2} + c_4\frac{n}{2^q}\,.
\end{aligned}$$

So scale 2^q is in $\mathcal{O}(n \log(n)^p)$. Multiplying by $\log_2(n)$ to consider all scales we get the desired result. □

F.1.3 MdFOCuS with dyadic update

The MdFOCuS Algorithm implementing this update of the set of candidates is presented in Algorithm 3. Combining Lemma 4 and 5 we get an overall expected complexity in $\mathcal{O}(n\log(n)^{p+1})$ complexity.

Algorithm 3 MdFOCuS(0) Dyadic algorithm

```
1:  **Input 1:** $p$-variate independent time series $\{x_t\}_{t=1,2,...}$
2:  **Input 2:** threshold $thrs$ and $\eta_1$ if known pre-change parameter
3:  **Input 3:** $q_{min}$ min power 2 scale at which we run *QuickHull*
4:  **Output:** stopping time $n$ and maximum likelihood change : $\hat{\tau}(.)$ or $\hat{\tau}(\eta_1)$
5:  $likelihoodRatio \leftarrow -\infty, \quad \mathcal{T} \leftarrow \emptyset, \quad n \leftarrow 0$
6:  while ($likelihoodRatio < thrs$) do
7:      $n \leftarrow n+1$
8:      $\mathcal{T} \leftarrow \mathcal{T} \cup \{n-1\}$
9:      $maxLikelihood \leftarrow \max_{\tau\in\mathcal{T}}\{\max_{\eta_1,\eta_2} l_{\tau,n}(\eta_1,\eta_2)\}$          ▷ $\eta_1$ is fixed in MdFOCuS0
10:     $likelihoodRatio \leftarrow maxLikelihood - \max_{\eta_1,\eta_2} l_{n,n}(\eta_1,\eta_2)$          ▷ $\eta_1$ is fixed in MdFOCuS0
11:     $q \leftarrow q_{min}$
12:     while $remainder(\frac{n-1}{2^q}) = 0$ do
13:         $\mathcal{T}' \leftarrow \{\tau \in \mathcal{T} | \tau > n - 2^q\}$
14:         $\mathcal{T}'' \leftarrow \text{QUICKHULL}(\{P(\tau)\}_{\tau\in\mathcal{T}'})$
15:         $\mathcal{T} \leftarrow \mathcal{T} \setminus (\mathcal{T}' \setminus \mathcal{T}'')$          ▷ discard vertices in $\mathcal{T}'$ not in $\mathcal{T}''$
16:         $q \leftarrow q+1$
17:     end while
18: end while
19: return $n$ and $likelihoodRatio$
```

We implemented Algorithm 3 also for a change in the mean of multi-dimensional Gaussian signal (see Table 1) in R/C++ using the qhull library (see link in the Supplementary Material).

We evaluate the empirical run times of Algorithm 3 with various values of q_{min} using the profiles of Appendix E.1. In Figure 14 we report the average run time as a function of the dyadic parameter q_{min} for MdFOCuS0 with dyadic update (that is for a known value of η_1) and values of q_{min} in $3,\ldots,12$. We observe a minimum run time for q_{min} in 6 for $p=1$, in 7 for $p=2$ and in 8 for $p=3$. As a consequence in the rest of the simulations we always set q_{min} to $p+5$.

Using the simulation framework of Section 4.3.1 we evaluated the run times of MdFOCuS0 with dyadic update for dimension $p=1$ to $p=3$ and compared it with run times of MdFOCuS. The average run times are presented in Figure 15. Dyadic MdFOCuS0 is slower than MdFOCuS0, about 2 times slower.

References

[1] Adam Olshen, E.S. Venkatraman, Robert Lucito, and Michael Wigler. Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics (Oxford, England)*, 5:557–72, 11 2004.

[2] Franck Picard, Stephane Robin, Marc Lavielle, Christian Vaisse, and Jean-Jacques Daudin. A statistical approach for array cgh data analysis. *BMC Bioinformatics*, 6:np, 2005.

[3] Jushan Bai and Pierre Perron. Computation and analysis of multiple structural-change. *Journal of Applied Econometrics*, 18, 01 2003.

[4] Alexander Aue, Lajos Horváth, Marie Hušková, and Piotr Kokoszka. Change-point monitoring in linear models. *The Econometrics Journal*, 9(3):373–403, 2006.

[5] Marcel Bosc, Fabrice Heitz, Jean-Paul Armspach, Izzie Namer, Daniel Gounot, and Lucien Rumbach. Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution. *NeuroImage 20(2)*, pages 643–656, 2003.

[6] Martin Staudacher, Stefan Telser, Anton Amann, Hartmann Hinterhuber, and Monika Ritsch-Marte. A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep. *Physica A-statistical Mechanics and Its Applications*, 349:582–596, 2005.

[7] Rakesh Malladi, Giridhar P. Kalamangalam, and Behnaam Aazhang. Online bayesian change point detection algorithms for segmentation of epileptic activity. *2013 Asilomar Conference on Signals, Systems and Computers*, pages 1833–1837, 2013.

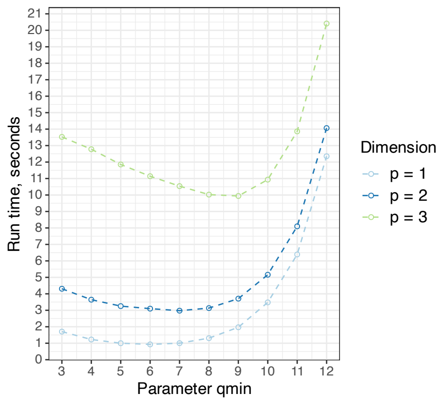


Figure 14: Run time of MdFOCuS0 with dyadic update as a function of q_{min} for p-variate time series with $n = 10^5$ data points in dimension $p = 1, 2$ and 3. We simulated 100 i.i.d. Gaussian data $\mathcal{N}_p(0, I_p)$ and report the average run time.

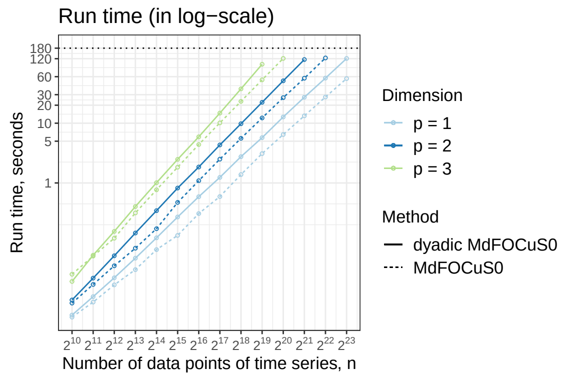


Figure 15: Run times in seconds of MdFOCuS0 (dotted) and MdFOCuS0 with dyadic update (full) both with $q_{min} = 5 + p$ in dimension $p = 1, 2$ and 3 using time series $x_{1:n}$ (without change, $x_t \sim \mathcal{N}_p(0, I_p)$ i.i.d). Run times are averaged over 100 data sets. We considered a maximum running time of 3 minutes for the algorithms. That is why we do not report run times for $p = 2$ and 3 for large signals.

[8] Jaxk Reeves, Jien Chen, Xiaolan L. Wang, Robert Lund, and Qi Qi Lu. A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meteorology and Climatology*, 46(6):900 – 915, dec 2007.

[9] Jean-François Ducré-Robitaille, Lucie A. Vincent, and Gilles Boulet. Comparison of techniques for detection of discontinuities in temperature series. *International Journal of Climatology*, 23, 2003.

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

[11] Itoh Naoki and Juergen Kurths. Change-point detection of climate time series by nonparametric method. *Lecture Notes in Engineering and Computer Science*, 2186, 10 2010.

[12] Claudie Beaulieu and Rebecca Killick. Distinguishing trends and shifts from memory in climate data. *Journal of Climate*, 31(23):9519 – 9543, 2018.

[13] Elena Andreou and Eric Ghysels. Detecting multiple breaks in financial market volatility dynamics. *Journal of Applied Econometrics*, 17(5):579–600, 2002.

[14] Piotr Fryzlewicz. Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6), dec 2014.

[15] Enric Galceran, Alexander Cunningham, Ryan Eustice, and Edwin Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots*, 41, 08 2017.

[16] David Rybach, Christian Gollan, Ralf Schluter, and Hermann Ney. Audio segmentation for speech recognition using segment features. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4197–4200, 2009.

[17] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *IEEE Transactions on Image Processing*, 14(3):294–307, 2005.

[18] Richard A. Davis, Thomas C.M. Lee, and Gabriel A. Rodriguez-Yam. Structural break estimation for nonstationary time series models. *Journal of the American Statistical Association*, 101:223–239, 2006.

[19] Ananth Ranganathan. Pliss: Labeling places using online changepoint detection. *Auton. Robots*, 32(4):351–368, may 2012.

[20] Sean Jewell, Paul Fearnhead, and Daniela Witten. Testing for a change in mean after changepoint detection., 2019.

[21] David Siegmund and ES Venkatraman. Using the generalized likelihood ratio statistic for sequential detection of a change-point. *The Annals of Statistics*, pages 255–271, 1995.

[22] Ivan E. Auger and Charles E. Lawrence. Algorithms for the optimal identification of segment neighborhoods. *Bulletin of Mathematical Biology*, 51(1):39–54, Jan 1989.

[23] Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[24] Andrew Jhon Scott and M Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, pages 507–512, 1974.

[25] Brad Jackson, Jeffrey D Scargle, David Barnes, Sundararajan Arabhi, Alina Alt, Peter Gioumousis, Elyus Gwin, Paungkaew Sangtrakulcharoen, Linda Tan, and Tun Tao Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105–108, 2005.

[26] Yi-Ching Yao. Estimation of a noisy discrete-time step function: Bayes and empirical bayes approaches. *The Annals of Statistics*, 12(4):1434–1447, 12 1984.

[27] Marc Lavielle and Eric Moulines. Least-squares estimation of an unknown number of shifts in a time series. *Journal of time series analysis*, 21(1):33–59, 2000.

[28] Emilie Lebarbier. Detecting multiple change-points in the mean of gaussian process by model selection. *Signal Processing*, 85:717–736, 04 2005.

[29] Damien Garreau and Sylvain Arlot. Consistent change-point detection with kernels, 2017.

[30] Weil R Lai, Mark D Johnson, Raju Kucherlapati, and Peter J Park. Comparative analysis of algorithms for identifying amplifications and deletions in array cgh data. *Bioinformatics*, 21(19):3763–3770, 2005.

[31] Arnaud Liehrmann, Guillem Rigaill, and Toby Dylan Hocking. Increased peak detection accuracy in over-dispersed chip-seq data with supervised segmentation models. *BMC bioinformatics*, 22(1):1–18, 2021.

[32] Guillem Rigaill. A pruned dynamic programming algorithm to recover the best segmentations with 1 to k_max change-points. *Journal de la Société Française de Statistique*, 156(4):180–205, 2015.

[33] Robert Maidstone, Toby Hocking, Guillem Rigaill, and Paul Fearnhead. On optimal multiple changepoint algorithms for large data. *Statistics and Computing*, 27(2):519–533, Mar 2017.

[34] Gaetano Romano, Idris Eckley, Paul Fearnhead, and Guillem Rigaill. Fast online changepoint detection via functional pruning cusum statistics, 2022.

[35] Vincent Runge. Is a finite intersection of balls covered by a finite union of balls in euclidean spaces? *Journal of Optimization Theory and Applications*, 187(2):431–447, 2020.

[36] Liudmila Pishchagina, Guillem Rigaill, and Vincent Runge. Geometric-based pruning rules for change point detection in multiple independent time series. *arXiv preprint arXiv:2306.09555*, 2023.

[37] Yudong Chen, Tengyao Wang, and Richard J Samworth. High-dimensional, multiscale online changepoint detection, 2022.

[38] Jaehyeok Shin, Aaditya Ramdas, and Alessandro Rinaldo. E-detectors: a nonparametric framework for sequential change detection. *arXiv preprint arXiv:2203.03532*, 2022.

[39] Haoyun Wang and Yao Xie. Sequential change-point detection: Computation versus statistical performance. *Wiley Interdisciplinary Reviews: Computational Statistics*, page e1628, 2022.

[40] Peter Eiauer and Peter Hackl. The use of mosums for quality control. *Technometrics*, 20(4):431–436, 1978.

[41] Alexander Meier, Claudia Kirch, and Haeran Cho. mosum: A package for moving sums in change-point analysis. *Journal of Statistical Software*, 97:1–42, 2021.

[42] Kes Ward, Giuseppe Dilillo, Idris Eckley, and Paul Fearnhead. Poisson-focus: An efficient online method for detecting count bursts with application to gamma ray burst detection, 2022.

[43] David Avis and David Bremner. How good are convex hull algorithms? In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 20–28, 1995.

[44] Franco P. Preparata and David E. Muller. Finding the intersection of n half-spaces in time o (n log n). *Theoretical Computer Science*, 8(1):45–55, 1979.

[45] Arne Brondsted. *An introduction to convex polytopes, by A. Brondsted. Pp 160. DM69. 1983. ISBN3-540-90722-X (Springer-Verlag)*. Springer New York, NY, 1983.

[46] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Texts and Monographs in Computer Science. Springer, 1985.

[47] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer Berlin, Heidelberg, 1987.

[48] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, dec 1996.

[49] Joseph O'Rourke. *Convex Hulls in Three Dimensions.*, page 101–154. Cambridge University Press, 2 edition, 1998.

[50] Raimund Seidel. Convex hull computations. In *Handbook of discrete and computational geometry*, pages 687–703. Chapman and Hall/CRC, 2017.

[51] Donald R. Chand and Sham S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17:78–86, 1970.

[52] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, feb 1977.

[53] Ben Kenwright. Convex hulls: Surface mapping onto a sphere, 2023.

[54] Nancy M. Amato, Michael T. Goodrich, and Edgar A. Ramos. Parallel algorithms for higher-dimensional convex hulls. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 683–694, 1994.

[55] Mikhail J. Atallah, Richard Cole, and Michael T. Goodrich. Cascading divide-and-conquer: A technique for designing parallel algorithms. *SIAM Journal on Computing*, 18(3):499–532, 1989.

[56] R. Miller and Q.F. Stout. Efficient parallel convex hull algorithms. *IEEE Transactions on Computers*, 37(12):1605–1618, 1988.

[57] Zakhar Kabluchko, Vladislav Vysotsky, and Dmitry Zaporozhets. Convex hulls of random walks: Expected number of faces and face probabilities., 2017.

[58] Victor S. Adamchik. On stirling numbers and euler sums. *Journal of Computational and Applied Mathematics*, 79:119–130, 1997.

[59] Kai Habel, Raoul Grasman, Robert B. Gramacy, Pavlo Mozharovskyi, and David C. Sterratt. *geometry: Mesh Generation and Surface Tessellation*, 2023. R package version 0.4.7.

[60] Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4(5):387–421, 1989.

[61] Avraham A. Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12, April 1987.

[62] Susan Hert and Stefan Schirra. 3D convex hulls. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6 edition, 2023.

[63] Wikipedia. Quickhull — Wikipedia, 2023. [Online; accessed 26-Oct-2023].

[64] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10(4):377–409, 1993.

[65] Alex Bresler. *nbastatR: R's interface to NBA data*, 2023. R package version 0.1.152.

[66] Wikipedia. David griffin — Wikipedia, 2023. [Online; accessed 27-Oct-2023].

[67] Kes Ward, Gaetano Romano, Idris Eckley, and Paul Fearnhead. A constant-per-iteration likelihood ratio test for online changepoint detection for exponential family models, 2023.

[68] Solène Thépaut. *Problèmes de clustering liés à la synchronie en écologie: estimation de rang effectif et détection de ruptures sur les arbres*. PhD thesis, Université Paris-Saclay (ComUE), 2019.

[69] Nancy Zhang and Siegmund David. A modified bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63:22–32, 04 2007.

[70] Yi-Ching Yao. Estimating the number of change-points via schwarz' criterion. *Statistics & Probability Letters*, 6(3):181–189, 1988.