Adaptive Gaussian Process Change Point Detection

Edoardo Caldarelli [*1] **Philippe Wenk** [2] **Stefan Bauer** [3] **Andreas Krause** [2]

Abstract

Detecting change points in time series, i.e., points in time at which some observed process suddenly changes, is a fundamental task that arises in many real-world applications, with consequences for safety and reliability. In this work, we propose ADAGA, a novel Gaussian process-based solution to this problem, that leverages a powerful heuristics we developed based on statistical hypothesis testing. In contrast to prior approaches, ADAGA adapts to changes both in mean and covariance structure of the temporal process. In extensive experiments, we show its versatility and applicability to different classes of change points, demonstrating that it is significantly more accurate than current state-of-the-art alternatives.

1. Introduction

Many real-world scenarios, such as quality control (Page, 1954), network analysis (Kurt et al., 2018) and finance (Lavielle & Teyssiere, 2007), posit the problem of detecting sudden changes in a data-generating process, especially in time series (Chandola et al., 2009; van den Burg & Williams, 2020). Finding such changes allows for isolating different patterns in the data, and has pivotal consequences in terms of safety and reliability of the predictive algorithms used, e.g., for risk or malfunction monitoring (Basseville et al., 1993; Skates et al., 2001; Galceran et al., 2017). In particular, the notion of *change point* (CP) refers to a point at which the hyperparameters of the model used to represent the data change suddenly.

CP detection in time series has been extensively studied over the years, e.g., by Scott & Knott (1974), Killick et al. (2012). In particular, a classical solution to the problem of CP detection is offered by Adams & MacKay (2007). They propose a Bayesian message-passing algorithm, called BOCPD, that relies on the concept of *run length*, i.e., time between subsequent CPs. In its original form, however, this algorithm assumes i.i.d. data between CPs. This is problematic in many time series applications, where temporal correlation between samples is the norm, as discussed by Saatçi et al. (2010). In particular, Saatçi et al. (2010) suggest to combine BOCPD with *Gaussian processes* (GPs), so as to model the time series. As a flexible and powerful tool to directly model the mapping from time to the signal (Williams & Rasmussen, 2006), GPs are well suited for this task.

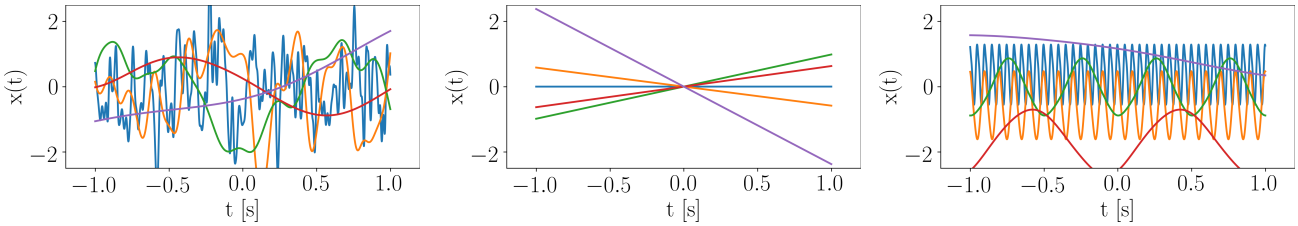
Although extremely flexible, these BOCPD-based detection schemes heavily rely on the choice of the prior hyperparameters, and this might worsen their performance, as discussed by Han et al. (2019). In particular, Han et al. (2019) observe that *statistical hypothesis testing* offers a more robust, yet effective way of dealing with CPs, as it allows for error thresholds to be derived. These thresholds can in turn be used to control false positive and false negative rates, which is crucial if the CP detection algorithm is to be used in a real-world scenario (Tartakovsky et al., 2014). These test-based approaches can be combined with deep learning methods, as done, e.g., by Chang et al. (2019), or with Gaussian processes. In particular, in the latter case, finding CPs translates into finding points at which the mean or covariance function of the underlying GP model changes. In particular, Keshavarz et al. (2018) devise a hypothesis testing procedure based on a generalized likelihood ratio test, to detect a single CP in the mean function of a Gaussian process with a fixed covariance function. Conversely, Han et al. (2019) propose a similar test, in combination with BOCPD, for a fixed mean function, which detects CPs in the covariance.

However, assuming either constant mean or a global covariance structure might reduce the class of anomalies that a CP detection algorithm can find, reducing in turn its applicability. We will show this in our experiments, where different types of CPs are considered. In particular, it will be clear that some phenomena, such as a gradual increase in the frequency of a sinusoidal signal, cannot be detected by an algorithm that only considers shifts in the mean.

In our work, we tackle the problem of GP-based CP detection from a heuristic point of view, designing a powerful

[*]This work was done while the author was at ETH Zurich [1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain [2]Department of Computer Science, ETH Zurich, Zurich, Switzerland [3]Department of Intelligent Systems, KTH, Stockholm, Sweden. Correspondence to: Edoardo Caldarelli <ecaldarelli@iri.upc.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.



(a) RBF kernel. (b) Linear kernel. (c) Periodic kernel.

Figure 1. Function samples drawn from some GP priors with different hyperparameters. The kernel function determines the shape of the function, and the class of CPs that can occur.

algorithm, based on statistical testing, which is able to detect CPs based on the GP hyperparameters *inferred from the data*, without relying on any assumptions of either the mean or the covariance being constant. In presence of a CP, our statistical test detects that a globally trained GP relies on unreliable hyperparameters, and partitions the dataset accordingly. For instance, a global GP might have an overly large noise variance, that actually accommodates for a shift in the mean of the series. Our algorithm is therefore particularly suitable for those real-world scenarios where the GP hyperparameters are not known *a priori*, and acts as an effective tool for validating the results of, e.g., a maximum-likelihood inference (Williams & Rasmussen, 2006; Clifton et al., 2012; Deisenroth et al., 2013). Furthermore, since every portion of the dataset between two subsequent CPs is standardized separately, the algorithm retrieves the best piece-wise constant mean function. Since it is set up in an online fashion and can easily be combined with different GP approximation methods, it can directly deal with streaming or big data problems, avoiding the well-known cubic complexity of GP regression. Thus, it efficiently and simultaneously handles several types of CPs, and can be applied to a wide range of practical scenarios, as we demonstrate empirically in our experiments, Section 4.

Contributions In summary, we

- devise a novel heuristics based on hypothesis testing and (approximate) GPs, to detect change points in our data-generating process,
- combine this framework with streaming data processing, to create a new algorithm, called ADAGA, that performs efficient CP detecion in the realistic scenario of ever-growing input domains,
- provide a publicly available implementation of ADAGA[1] and use it to in the context of CP detection, with different types of CPs being processed. In all cases, ADAGA substantially outperforms the state-of-the-art algorithms, in terms of accuracy and versatility.

[1]Code available at `https://github.com/lasgroup/adaga`. The implementation uses `scipy` (Virtanen et al., 2020), `tensorflow` (Abadi et al., 2016) and `gpflow` (De G. Matthews et al., 2017).

2. Background: GP Regression

In this section, we recall the main preliminaries of GP regression that form the basis of our work. An extensive presentation of these topics can be found in Williams & Rasmussen (2006).

GP Regression For notational simplicity, we consider in this section the task of 1-dimensional GP regression. Given a vector of input points $\mathbf{t} \in \mathbb{R}^n$, we aim at learning a function $x\colon \mathbb{R} \to \mathbb{R}$ at these points, from a set of n noisy observations, which we denote by $\mathbf{y} \in \mathbb{R}^n$. We call the tuple $(\mathbf{t}, \mathbf{y})$ *dataset*. Moreover, we denote a datapoint in the dataset as the tuple (t_j, y_j), for $j \in \{1, \dots, n\}$. The approach easily extends to multidimensional input, where the domain is a subset of $\mathbb{R}^d$, $d > 1$, and the element t_i in a datapoint is a vector in $\mathbb{R}^d$.

A kernel function $k(\cdot, \cdot)$, parameterized with a set of hyperparameters $\boldsymbol{\phi}$, and a mean function $\mu(\cdot)$ allow us to define a Gaussian prior distribution over the functions among which the function $x(\cdot)$ is sampled:

$$p(\mathbf{x}(\mathbf{t})|\boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{t}), \mathbf{C}). \tag{1}$$

Each element in the covariance matrix $\mathbf{C}$ is given by

$$\mathbf{C}_{i,j} = k(t_i, t_j). \tag{2}$$

If the kernel depends only on the distance between input points, then it is called *stationary*.

W.l.o.g., we choose $\mu(\cdot)$ to be 0. Moreover, we assume that the observations are corrupted by additive, zero-mean Gaussian noise with variance σ^2. This results in the Gaussian likelihood model

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\phi}, \sigma^2) = \mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I}). \tag{3}$$

From this, we can derive the Gaussian posterior of the function values, given the noisy observations, $p(\mathbf{x}|\mathbf{y}, \boldsymbol{\phi}, \sigma^2)$, with mean and covariance

$$\boldsymbol{\mu}_{post} = \mathbf{C}(\mathbf{C} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, \tag{4}$$

$$\mathbf{C}_{post} = \sigma^2(\mathbf{C} + \sigma^2\mathbf{I})^{-1}\mathbf{C}. \tag{5}$$

The prior hyperparameters and the noise variance are inferred by maximizing the marginal log-likelihood of the

observations

$$2\log p(\mathbf{y}|\boldsymbol{\phi},\sigma^2) = -\mathbf{y}^T\left(\mathbf{C}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{y} - \log\left[\det\left(\mathbf{C}+\sigma^2\mathbf{I}\right)\right] - n\log 2\pi. \quad (6)$$

Approximating the Kernel Due to matrix inversions in Equations (4), (5) and (6), standard GP regression scales cubically in the number of observations. To improve the computational complexity, the most common approach is to replace the covariance matrix with a low-rank approximation. In this paper, we consider two families of approximations, namely *inducing points* (Snelson & Ghahramani, 2006; Titsias, 2009; Hensman et al., 2013) and *feature approximations*.

Inducing points are a small set of pseudo-inputs summarizing the whole dataset. In this case, we assume that the matrix in Equation (2) factorizes as $\mathbf{C}_{n,m}\mathbf{C}_{m,m}^{-1}\mathbf{C}_{m,n}$. $\mathbf{C}_{n,m}$ represents the correlation between inputs and the inducing points, $\mathbf{C}_{m,m}$ the correlation among the inducing points themselves, and $\mathbf{C}_{m,n} = \mathbf{C}_{n,m}^T$. The optimal location of the inducing points, along with the usual optimization parameters of GP regression, is learned by maximizing the well-known SGPR bound on the marginal log-likelihood of the observations as presented by Titsias (2009).

On the other hand, feature approximation schemes (Williams & Seeger, 2001; Rahimi & Recht, 2008; Mutny & Krause, 2018) approximate the value of the kernel function at timesteps t_i, t_j as the inner product of two finite-dimensional vectors. These vectors form the column of the the so-called *feature embedding matrix* $\mathbf{\Phi}$, meaning that the covariance matrix factorizes as $\mathbf{\Phi}^T\mathbf{\Phi}$. We can observe that this approximation is formally equivalent to the inducing points' one, when we substitute $\mathbf{C}_{m,m}$ with the identity matrix, and $\mathbf{C}_{m,n}$ with $\mathbf{\Phi}$. Although there exist several methods for computing feature embeddings, we consider the deterministic and provably accurate scheme provided by quadrature Fourier features (QFFs), presented by Mutny & Krause (2018).

Partitioning Schemes Instead of approximating the kernel, an additional way to reduce the complexity of GP regression consists of training multiple local models. For this purpose, we cluster the datapoints into p subsets $\mathcal{P}_1,\dots,\mathcal{P}_p \subseteq \mathcal{D}$, creating a *partition* of the dataset. This approach is particularly suitable for streaming scenarios (Nguyen-Tuong et al., 2009; Stork & Stoyanov, 2020), as it allows to control the size of the GPs' supports, and prevents the runtime from becoming prohibitive, even with an ever-increasing dataset.

3. Adaptive Streaming GP Regression

In this section, we describe ADAGA, our algorithm for GP-based change point detection. Unless explicitly stated, all the theoretical results are our novel contributions and proven in the supplementary material, Sections E, F.

3.1. CPs and Kernel Function

Before presenting our algorithm in detail, we can observe that the notion of CP is strictly related to the kernel function used, for a GP-based CP detection scheme. As shown in Figure 1, the choice of the kernel dictates the shape of the functions that can be drawn from the prior distribution. This in turn means that the type of CPs that can be detected by a GP-based algorithm is determined by the process that is used as prior model. For instance, a change in a linear trend can be interpreted as a change in the variance of a linear kernel, $k(t_i,t_j) = \sigma^2_{linear}t_it_j$, whereas a change in the variance and smoothness of the data can be related to the hyperparameters of a radial-basis-function (RBF) kernel, $k(t_i,t_j) = \sigma^2_{RBF}\exp\left\{-|t_i-t_j|^2/2l^2\right\}$.

3.2. Streaming Problem Setup

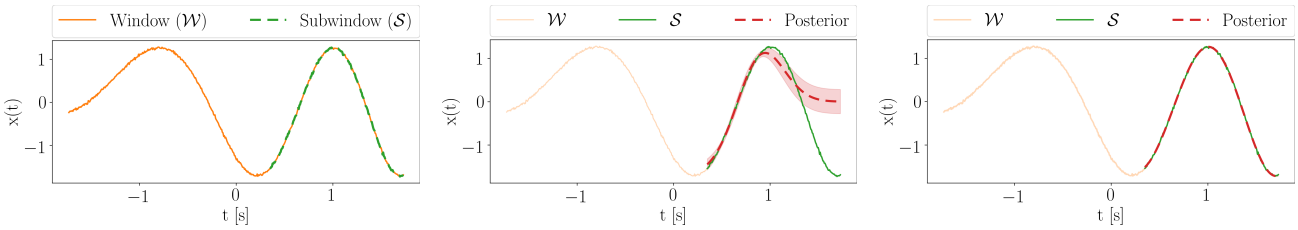
Having clarified the notion of CP within a GP-based framework, we can introduce the fundamental concepts that form the basis of our algorithm, ADAGA, and then characterize them in two well-known streaming scenarios, described by Keshavarz et al. (2018).

Windows and Subwindows ADAGA is designed to work with a streaming data source. That is, we consider a sequence of datapoints (t_i, y_i) to be sampled for a potentially infinite amount of time. We allow the datapoints to be sampled in batches with arbitrary size. At this stage, we do not make any assumptions on the location of the datapoints with respect to each other. As ADAGA gathers new datapoints, it creates and updates a partition of the dataset $\mathcal{D}$. We denote by *windows* the subsets $\mathcal{W}_1,\dots,\mathcal{W}_p$ belonging to this partition. In addition to the standard properties of a partition, all the windows created by ADAGA contain datapoints that are adjacent, w.r.t. to the order in which they are *sampled*. This is encoded in the following condition:

$$\forall i \in \{1,\dots,p\},\ \exists\, m, M \in \{1,\dots,n\}\ \text{ s. t. } \mathcal{W}_i = \{(t_w,y_w)\in\mathcal{D}|\ m\le w\le M\}. \quad (7)$$

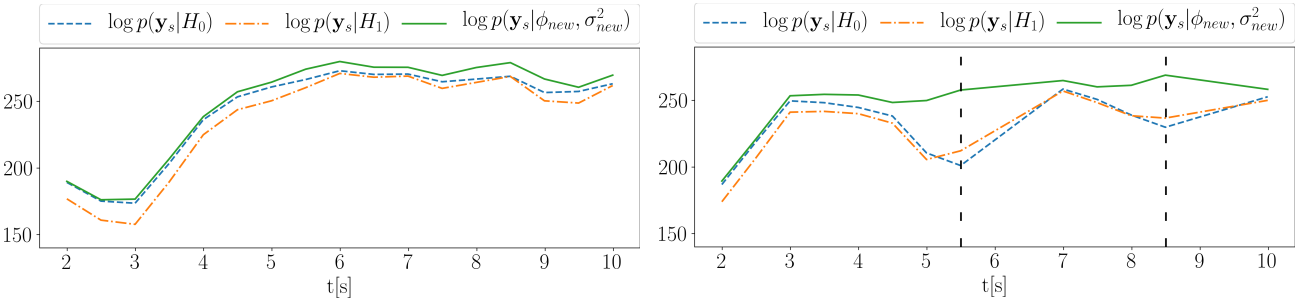
We can observe that the general definition of a window is independent of the *location* of the sampled points in the function's domain. The partition is created dynamically. During streaming, the batches are appended to the same window, until it is found to contain a CP. Such a window is then said to be *spoiled*. Thus, a reset is performed, and a new window is created with the newly arriving datapoints.

In the next subsection, we design a criterion that implicitly guarantees that the current window is not spoiled at each step of the streaming, determining the size of each window adaptively w.r.t. the dynamics of the process. This criterion is based on the definition of *subwindow*, $\mathcal{S}$, which is a



(a) Original data. (b) $\mathcal{M}_c$, trained on $\mathcal{W}$. (c) $\mathcal{M}_n$, trained on $\mathcal{S}$.

Figure 2. Figure 2a shows all datapoints in the current window $\mathcal{W}$ and the corresponding subwindow $\mathcal{S}$ of one step of ADAGA, for temporal streaming. In red, we show mean and standard deviations of the posterior of $\mathcal{M}_c$ and $\mathcal{M}_n$, the two GP models considered by the statistical test. Clearly, a CP should be detected, as training on $\mathcal{S}$ leads to a more accurate posterior than $\mathcal{W}$. $\mathcal{W}$ is spoiled.



(a) $x(t) = \sin(t)$ (b) $x(t) = \sin(t^{1.5})$

Figure 3. Log-likelihoods involved in our test, on 2 different functions, processed with ADAGA, using an RBF kernel with 10 inducing points and 1000 observations in $[0, 10]$. The function on the left can be modeled without the need of splitting its domain. However, the likelihood based on just the subwindow is steadily larger than the null likelihood, which is not desirable. This is due to $\mathcal{M}_n$ being trained via maximum marginal likelihood. Conversely, our PoE alternative likelihood is not larger than the null likelihood, unless the current window is becoming spoiled (at $5.5s$ and $8.5s$ in the figure on the right), and a CP is detected. This is because the function on the right is gradually accelerating, and a single, global RBF kernel cannot model it properly.

subset of a window fulfilling Equation (7). $\mathcal{S}$ is placed on the last part of the window, so as to include the most recent data. Contrary to the size of the windows, the size of the subwindows $|\mathcal{S}|$ must be fixed throughout the whole execution of the algorithm. While $|\mathcal{S}|$ can be chosen to be a hyperparameter, we can also leverage prior knowledge about the data distribution to derive a theoretically grounded size, as shown in the supplementary material, Section C. In the same section, we also show how a wrong subwindow's size affects the CP detection. We can observe that the definition of subwindow requires that, at each step of the streaming,

$$|\mathcal{W}_i| \geq |\mathcal{S}|. \tag{8}$$

This automatically prevents ADAGA from working with excessively small windows, making the optimization less prone to overfitting.

Increasing-domain Streaming The type of streaming we consider in our work is the *increasing-domain streaming*, which arises when considering, e.g., time series data. The key assumption is that the streaming follows the ordering of the points in the domain. For 1-dimensional data, this assumption can be formalized as follows:

$$\forall (t_h, y_h), (t_k, y_k) \in \mathcal{D}, h < k \Rightarrow t_h < t_k. \tag{9}$$

This means that the size of the domain of $x(\cdot)$ increases during streaming, while the sampling frequency stays constant. If the domain of the function is multidimensional, we can consider a suitable ordering for the points (e.g., a spatial ordering). The points belonging to a window, and the ones in a subwindow, are adjacent within the function's domain.

Fixed-domain Streaming Another type of streaming is *fixed-domain* streaming, where the function's domain is fixed, and the points are sampled from it, leading to a denser dataset over time. In this case, a window comprises the most recently sampled points, and ADAGA can be used to detect abrupt changes in the process from which the function is drawn, occurring during the streaming. However, due to the limited practical applicability of this scenario, we do not investigate it in this work.

3.3. Partitioning Heuristics

ADAGA's partitioning strategy relies on detecting when a window has grown excessively, so as to include a CP. Intuitively, we can observe that a window is spoiled if introducing a local GP on the subwindow improves the regression. More specifically, the following set of hypotheses allows us to quantify the beneficial effect of a GP trained on the sub-

window only. We will first give a high-level description and then state them more formally in Definition 3.1. Note that our alternative hypothesis is heurisics-based by definition. Thus, it leads to a surrogate likelihood function. This should not be confused with the true underlying data-generating process, which is not explicitly modeled.

$\mathbf{H_0}$: The null hypothesis assumes that the function values in the subwindow come from the same observational model as the rest of the window.

$\mathbf{H_1}$: The alternative hypothesis assumes that the function values in the subwindow come from the superposition of two GP experts: $\mathcal{M}_c$, potentially spoiled, which has the same parameters as the rest of the window, and $\mathcal{M}_n$, potentially much better, whose parameters are inferred from the subwindow only. To prevent the likelihood from collapsing to the likelihood of a single component, the two likelihoods are multiplied together (instead of being, e.g., summed). The resulting model is a combination of a mixture-of-experts (Masoudnia & Ebrahimpour, 2014) and a product-of-experts heuristics (Hinton, 2002). The density of the subwindow is a product of experts, since the likelihood of both GP experts is multiplied together. The overall data is modelled via a mixture of experts. While we use the product of experts likelihood on the subwindow, the first part of the data is modelled via the first GP only.

Definition 3.1. Let $\boldsymbol{\phi}_{H_0}$ and $\sigma^2_{H_0}$ be the GP hyperparameters and noise variance learned from the whole current window. The null hypothesis H_0 is that the marginal likelihood of the observations $\mathbf{y}_s$ in the subwindow is

$$p(\mathbf{y}_s|H_0) = p(\mathbf{y}_s|\mathbf{t}_s, \boldsymbol{\phi}_{H_0}, \sigma^2_{H_0}).$$

Moreover, let $\boldsymbol{\phi}_{new}$ and σ^2_{new} be the GP hyperparameters and noise variance learned from the subwindow, and $Z_1 \neq 0$ be a proper normalization constant. The alternative hypothesis H_1 is that the marginal likelihood of the observations $\mathbf{y}_s$ in the subwindow is

$$p(\mathbf{y}_s|H_1) = \frac{p(\mathbf{y}_s|\mathbf{t}_s, \boldsymbol{\phi}_{H_0}, \sigma^2_{H_0}) p(\mathbf{y}_s|\mathbf{t}_s, \boldsymbol{\phi}_{new}, \sigma^2_{new})}{Z_1}.$$

On the subwindow, we now have two different likelihood models for the same points $\mathbf{y}_s$. Thus, we can compare them with a *likelihood ratio test*. In particular, we can define our test statistic as follows:

Definition 3.2. Let Z_{new} be the normalization constant for the GP expert whose hyperparameters are learned from the subwindow only, and let $\mathbf{V}_{new}$ be its covariance matrix. Given null and alternative hypotheses, the likelihood ratio is

$$\mathcal{R} = 2 \ln \frac{p(\mathbf{y}_s|H_1)}{p(\mathbf{y}_s|H_0)} = 2 \ln \frac{1}{Z_1} - 2 \ln Z_{new} - \mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s.$$

For a given threshold value $\mathcal{T}$ specified later, we accept the alternative hypothesis (i.e., we say that the current window is spoiled) if

$$\mathcal{R} \geq \mathcal{T}. \tag{10}$$

The resulting algorithm is summarized in Algorithm 1. The threshold values used there are presented in the next subsection. As presented in Algorithm 1, when a window is found to be spoiled, a reset is performed. The end of the current window is located $|S|$ points before the current end, and so is the beginning of the new window, which comprises the last $|\mathcal{S}|$ points seen. This choice is motivated by the observation that, if a CP has occurred, these points are known to belong to a new data-generating process. The initial size of a window is set to $2|\mathcal{S}|$, so that, if a reset occurs, the condition described in Equation (8) holds. Moreover, Figure 2 qualitatively compares the posteriors of $\mathcal{M}_c$ and $\mathcal{M}_n$, on a spoiled window.

Algorithm 1 ADAGA

Input: $|\mathcal{S}|$, $\mathbf{t}$, $\mathbf{y}$, δ.
Output: p disjoint subsets of $(\mathbf{t}, \mathbf{y})$.
while $\exists(\mathbf{t}_{new}, \mathbf{y}_{new})$ **do**
  $\mathbf{t}_{window} \Leftarrow [\mathbf{t}_{window}, \mathbf{t}_{new}]$.
  $\mathbf{y}_{window} \Leftarrow [\mathbf{y}_{window}, \mathbf{y}_{new}]$.
  if $|(\mathbf{t}_{window}, \mathbf{y}_{window})| < 2|\mathcal{S}|$ **then**
    continue
  end if
  Standardize $\mathbf{t}_{window}, \mathbf{y}_{window}$.
  Train a GP $\mathcal{M}_c$ on $\mathbf{t}_{window},\ \mathbf{y}_{window}$.
  Initialize $\mathbf{t}_s,\ \mathbf{y}_s$ with the last $|\mathcal{S}|$ points in the window.
  Train a GP $\mathcal{M}_n$ on $\mathbf{t}_s,\ \mathbf{y}_s$.
  Compute $\mathcal{R}$, using $\mathcal{M}_c$ and $\mathcal{M}_n$.
  Compute the thresholds $\mathcal{T}_I,\ \mathcal{T}_{II}$ using δ.
  Perform the statistical test using $\mathcal{R},\ \mathcal{T}_I,\ \mathcal{T}_{II}$.
  if the test rejects H_0 **then**
    Save $(\mathbf{t}_{window}, \mathbf{y}_{window})$ without the last $|\mathcal{S}|$ datapoints.
    $\mathbf{t}_{window} \Leftarrow$ Last $|\mathcal{S}|$ points in $\mathbf{t}_{window}$.
    $\mathbf{y}_{window} \Leftarrow$ Last $|\mathcal{S}|$ points in $\mathbf{y}_{window}$.
  end if
end while

3.4. Thresholds

The threshold used in Equation (10) needs to be chosen so as to bound the probability of making errors, that is, accepting the alternative hypothesis when the null one holds and vice versa.

Controlling Type I Errors For the so-called *type I errors*, we are interested in bounding the probability $\mathbb{P}[\mathcal{R} \geq \mathcal{T}_I|H_0]$, where $\mathcal{T}_I$ is the alternative hypothesis' acceptance threshold. The following theorem provides us with a criterion for choosing a threshold that controls the type I error probability.

Theorem 3.3. *Let $\mathbf{V}_{H_0}$ be the covariance matrix of the likelihood in the null hypothesis. Moreover, let $\delta \in (0,1)$, λ_{i,H_0} be the* i*-th eigenvalue of the matrix $\mathbf{V}_{H_0}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_0}^{1/2}$, and $\mu_{H_0} = \mathbb{E}[\mathcal{R}|H_0]$. Setting*

$$\mathcal{T}_I = \mu_{H_0} + \max\{\mathcal{C}_{0,H_0}, \mathcal{C}_{1,H_0}\},$$

where

$$\mathcal{C}_{0,H_0} = \sqrt{8\ln(1/\delta)\sum_i \lambda_{i,H_0}^2},$$
$$\mathcal{C}_{1,H_0} = 8\ln(1/\delta)\max_i\{\lambda_{i,H_0}\},$$

guarantees a type I error probability of at most δ.

Controlling Type II Errors A similar procedure can be followed to determine the acceptance threshold for controlling the so called *type II errors*. In this case, we are interested in bounding the probability $\mathbb{P}[\mathcal{R} \leq \mathcal{T}_{II}|H_1]$, where $\mathcal{T}_{II}$ is the alternative hypothesis' acceptance threshold. We can derive the following criterion for choosing a threshold that controls the type II error probability.

Theorem 3.4. *Let $\mathbf{V}_{H_1}$ be the covariance matrix of the likelihood in the alternative hypothesis, that is, according to the PoE formulation,*

$$\mathbf{V}_{H_1} = \left(\mathbf{V}_{H_0}^{-1} + \mathbf{V}_{new}^{-1}\right)^{-1}.$$

Moreover, let $\delta \in (0,1)$, λ_{i,H_1} be the i*-th eigenvalue of the matrix $\mathbf{V}_{H_1}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_1}^{1/2}$, and $\mu_{H_1} = \mathbb{E}[\mathcal{R}|H_1]$. Setting*

$$\mathcal{T}_{II} = \mu_{H_1} - \max\{\mathcal{C}_{0,H_1}, \mathcal{C}_{1,H_1}\},$$

where

$$\mathcal{C}_{0,H_1} = \sqrt{8\ln(1/\delta)\sum_i \lambda_{i,H_1}^2},$$
$$\mathcal{C}_{1,H_1} = 8\ln(1/\delta)\max_i\{\lambda_{i,H_1}\},$$

guarantees a type II error probability of at most δ.

Combining the Thresholds We can observe that, if $\mathcal{T}_I \leq \mathcal{T}_{II}$, we have thresholds that guarantee low error probability for both type I and type II (namely, the ones between $\mathcal{T}_I$ and $\mathcal{T}_{II}$). Thus, in the test, we actually compare the test statistic against the threshold $\mathcal{T}_I$, only when such condition holds.

Further Discussion on the PoE Heuristics Note that in the alternative hypothesis, the PoE is taken between the marginal likelihoods $p(\mathbf{y}_s|\mathbf{t}_s)$, not on the level of priors. Thus, it can not be directly expressed as a structural change in the kernel. Intuitively, we might be tempted to get rid of the PoE in H_1 completely and just consider the newly trained hyperparameters. As we found in our preliminary experiments, the new hyperparameters can be prone to overfitting (see an example in Figure 3). This is no surprise, as the hyperparameters are trained by maximizing the likelihood. While there exist alternatives to this training scheme, e.g. the work of Vehtari et al. (2017), we found that our choice of H_1 solves this problem most reliably in practice. In principle, a perfect likelihood ratio test should be able to catch these issues and absorb it into its type I and type II error probabilities and the related thresholds. However, up to our knowledge, it is currently an open question how to analytically calculate the quantities in question, and loose thresholds might prevent the statistical test from working at all. Fortunately, as we see in Definition 3.2, the PoE in H_1 allows for the H_0 density to cancel and we thus obtain positive definite matrices $\mathbf{V}_{H_0}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_0}^{1/2}$ and $\mathbf{V}_{H_1}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_1}^{1/2}$, which the thresholds depend on. With these matrices, we can deploy efficient, subexponential bounds. Thus, to facilitate a theoretical analysis, it is crucial that the alternative hypothesis does not incorporate soft transitions, as e.g. done in Lloyd et al. (2014), and has a clearly defined change point at the beginning of the subwindow, in contrast to e.g. Han et al. (2019), as this would not allow for similarly tight bounds as we obtain in Theorems 3.3 and 3.4, whose tightness is crucial for empirical performance. As we shall see later, the resolution of change point locations can always be adjusted by choosing the batch size of our algorithm.

3.5. Efficient Implementation of ADAGA

In this part, we discuss some details related to the statistical test. These are relevant in order to get an efficient implementation of ADAGA, when the low-rank factorizations of the kernel matrix, described in Section 2, are used to overcome the cubic complexity of naive GP regression.

Simplified Test Since the expectation of a constant is the constant itself, we have the following Lemma.

Lemma 3.5. *Let $Tr[\cdot]$ be the trace operator, $k \in \{0,1\}$, and $\mathbf{V}_{new}$ be the covariance of $\mathcal{M}_n$, the expert trained on the subwindow. Let $\hat{\mu}_{H_k} = \mathbb{E}\left[\mathbf{y}^T\mathbf{V}_{new}^{-1}\mathbf{y}\right]$. Then,*

$$\hat{\mu}_{H_k} = Tr\left[\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}\right].$$

This can be used to simplify the statistical test, as it is no longer required to compute the normalization constants. Thus, instead of considering $\mathcal{R}$ as in Definition 3.2, we can use $-\mathbf{y}^T\mathbf{V}_{new}^{-1}\mathbf{y}$ as test statistic, and replace the complete μ in the thresholds with $-\hat{\mu}$ from Lemma 3.5.

Eigenvalues for the Thresholds In addition to the previously outlined results, we can now observe that the quantities involved in the thresholds do not require an explicit calculation of the eigenvalues. In particular, let us first assume that the matrices $\mathbf{V}_{H_0}$, $\mathbf{V}_{H_1}$ and $\mathbf{V}_{new}^{-1}$ can be computed efficiently. We have that the following lemmas hold.

Table 1. Average F-1 scores (with standard deviation) of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across three *synthetic* datasets (for 10 noisy realizations). A margin of 5 points was used. Bold values indicate the highest average score for the dataset. Last column shows the overall average score across all noisy realizations of the datasets. Precision, recall, and a visualization of the CPs can be found in the supplementary material.

ALGORITHM	MEAN SHIFT DATA	VARIANCE SHIFT DATA	PERIODICITY SHIFT DATA	AVERAGE
ADAGA (QFFs, RBF)	0.68 ± 0.12	$\mathbf{0.75 \pm 0.24}$	0.53 ± 0.2	0.65
ADAGA (IPs, RBF)	$\mathbf{1.0} \pm 0$	0.6 ± 0.2	0.58 ± 0.15	**0.73**
ADAGA (IPs, Matern52)	$\mathbf{1.0} \pm 0$	0.6 ± 0.2	0.59 ± 0.19	**0.73**
ADAGA (IPs, RQ)	$\mathbf{1.0} \pm 0$	0.6 ± 0.2	$\mathbf{0.6} \pm 0.18$	**0.73**
ADAGA (IPs, periodic)	–	–	0.29 ± 0	0.29
BINSEG (mean)	0.67 ± 0	0.5 ± 0	0.4 ± 0.04	0.52
BINSEG (mean & var)	0.67 ± 0	0.32 ± 0.02	0.36 ± 0.1	0.45
PELT (mean)	0.67 ± 0	0.51 ± 0.1	0.34 ± 0.04	0.51
PELT (mean & var)	0.53 ± 0.08	0.55 ± 0.12	0.38 ± 0.08	0.49
BOCPD	0.71 ± 0.09	0.65 ± 0.12	0.47 ± 0.12	0.61
RBOCPDMS	0.31 ± 0.02	0.43 ± 0.08	0.52 ± 0.17	0.42
GPTS-CP (RQ+const)	0.94 ± 0.15	0.59 ± 0.18	0.5 ± 0	0.68
ZERO	0.5 ± 0	0.5 ± 0	0.5 ± 0	0.5

Table 2. F-1 scores of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across six *real-world* datasets. A margin of 5 points was used. Bold values indicate the highest score for the dataset. Last column shows the average score across the datasets. Precision, recall, and a visualization of the CPs can be found in the supplementary material.

ALGORITHM	RUN LOG	BUSINV	OZONE	GDP IRAN	GDP ARGENTINA	GDP JAPAN	AVERAGE
ADAGA (exact, linear)	0.57	**0.77**	–	–	–	–	0.67
ADAGA (IPs, linear)	0.60	0.63	–	–	–	–	0.62
ADAGA (QFFs, RBF)	–	–	0.97	**0.87**	0.82	**0.89**	**0.89**
ADAGA (IPs, RBF)	–	–	0.78	0.80	0.89	0.62	0.77
ADAGA (IPs, Matern52)	–	–	0.97	0.80	0.82	**0.89**	0.87
ADAGA (IPs, RQ)	–	–	0.97	0.80	0.82	0.62	0.8
BINSEG (mean)	0.43	0.37	0.65	0.49	0.89	0.62	0.57
BINSEG (mean & var)	0.35	0.24	0.56	0.39	0.8	0.57	0.49
PELT (mean)	0.31	0.37	**1.0**	0.49	0.89	0.62	0.61
PELT (mean & var)	0.45	0.20	0.60	0.44	0.67	0.50	0.48
BOCPD	0.52	0.27	0.75	0.39	0.80	0.80	0.59
RBOCPDMS	0.42	0.27	0.78	0.49	0.58	0.47	0.50
GPTS-CP (linear+const)	**0.84**	0.62	–	–	–	–	0.73
GPTS-CP (RQ+const)	–	–	0.65	**0.87**	**0.95**	0.66	0.78
ZERO	0.45	0.59	0.72	0.65	0.82	**0.89**	0.69

Lemma 3.6. *The eigenvalues of* $\mathbf{V}_{H_0}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_0}^{1/2}$ *resp.* $\mathbf{V}_{H_1}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_1}^{1/2}$ *are the same as the ones of* $\mathbf{V}_{H_0}\mathbf{V}_{new}^{-1}$ *resp.* $\mathbf{V}_{H_1}\mathbf{V}_{new}^{-1}$.

Lemma 3.7. *Let* $\boldsymbol{\lambda}_{H_0}$ *and* $\boldsymbol{\lambda}_{H_1}$ *be the vectors of the eigenvalues of* $\mathbf{V}_{H_0}\mathbf{V}_{new}^{-1}$ *resp.* $\mathbf{V}_{H_1}\mathbf{V}_{new}^{-1}$. *Moreover, let* $k \in \{0, 1\}$. *Then,*

$$\sum_i \lambda_{i,H_k}^2 = ||\boldsymbol{\lambda}_{H_k}||_2^2 = Tr\left[\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}\right].$$

Moreover, by the definition of infinity norm of a vector, $||\cdot||_\infty$, we have

$$\max_i \{\lambda_{i,H_k}\} = ||\boldsymbol{\lambda}_{H_k}||_\infty, \tag{11}$$

which can be computed efficiently via the Arnoldi method (Lehoucq et al., 1998).

We are then left with the question of whether the $|\mathcal{S}| \times |\mathcal{S}|$ matrices $\mathbf{V}_{H_0}$, $\mathbf{V}_{new}^{-1}$ and $\mathbf{V}_{H_1}$ can be computed efficiently when using our low-rank approximations of the covariance matrix, described in Section 2. By applying the well known *Woodbury identity*, a.k.a. *matrix inversion lemma* (Woodbury, 1950), we obtain the following proposition.

Proposition 3.8. *Let* m *be the number of inducing points (or QFFs) used. Then, the matrices* $\mathbf{V}_{H_0}$, $\mathbf{V}_{H_1}$ *and* $\mathbf{V}_{new}^{-1}$ *can be computed in* $\mathcal{O}(m^3)$.

An extensive discussion on the computational complexity of ADAGA can be found in the supplementary material, Section D.

4. Experiments

This section reports the empirical results obtained using ADAGA in the context of CP detection in time series. This scenario allows us to test ADAGA with different kernels and both the approximation schemes described in Section 2,

Table 3. Average delay (with standard deviation) of ADAGA for CP detection, across three *synthetic* datasets (for 10 noisy realizations). A margin of 5 points was used.

ALGORITHM	MEAN SHIFT DATA	VARIANCE SHIFT DATA	PERIODICITY SHIFT DATA
ADAGA (QFFs, RBF)	15.08 ± 2.02	11.86 ± 1.6	16.6 ± 3.44
ADAGA (IPs, RBF)	15.25 ± 0.89	10.8 ± 0.75	14.33 ± 3.4
ADAGA (IPs, Matern52)	15 ± 0	11.0 ± 0.89	15.25 ± 3.83
ADAGA (IPs, RQ)	15 ± 0	10.8 ± 0.75	16 ± 4.06
ADAGA (IPs, periodic)	–	–	NA

and, most importantly, showcase the benefits offered by our algorithm. All the experiments were performed on a custom laptop (Macbook Pro 2019). The experiments with CP detection consist of two parts. Firstly, we use some synthetic datasets. Secondly, we consider some real-world datasets described by van den Burg & Williams (2020). The goal of these experiments is to show that a suitable kernel choice allows ADAGA to flexibly detect a broad set of different types of CPs, outperforming its competitors. The extended empirical results can be found in the supplementary material, Sections G, H.

Synthetic Time Series Our three synthetic series have known CPs. In particular, we consider a series with two shifts in mean, one with two shifts in the noise variance, and one with two shifts in the periodicity of the signal. A detailed description of these series can be found in the supplementary material. For each series, we consider 10 different noisy realizations. These are processed with the RBF kernel, both with 30 QFFs and 10 inducing points; the Matern52 kernel, with 10 inducing points; the rational quadratic (RQ) kernel, with 10 inducing points. In addition, the last series is processed with a periodic kernel (MacKay, 1998).

Real-world Time Series Secondly, we use some of the real-world benchmarks described by van den Burg & Williams (2020). In our work, we choose the following sets, whose detailed description is reported in the supplementary material: Run Log, Business Inventories, Ozone, Iran's GDP, Argentina's GDP, Japan's GDP. Since the Run Log and the Business Inventories datasets consist of varying linear trends, we use a linear kernel in order to process these series. This kernel is tested both with 10 inducing points, and using $\sigma_{linear} t_i$ as the exact 1-dimensional embedding for the i-th timestep. In all the other cases, we used 3 related types of kernels: the RBF kernel, both with 30 QFFs and 10 inducing points; the Matern52 kernel, with 10 inducing points; the rational quadratic (RQ) kernel, with 10 inducing points. As described by van den Burg & Williams (2020), the datasets were annotated by experts. This allows us to quantitatively evaluate the performance of the algorithms of interest against a ground truth.

F-1 Score As described by van den Burg & Williams (2020), the exact location of a CP can be affected by some degree of arbitrariness. This can be observed in the annotations of the real-world time series, where the experts rarely agree on the exact location of a CP. To overcome this issue, as proposed by van den Burg & Williams (2020), we choose a modified *F-1 score* as evaluation metric. Basically, a CP is considered correctly identified if the algorithm chooses any candidate within a certain margin around a true CP. As reported in the supplementary material, choosing a margin of 0 leads to meaningless results. Thus, a slightly larger margin is needed. We choose a margin of 5 points, but report a margin of 10 points in the supplementary material to demonstrate robustness of the evaluation scheme.

Benchmarking For both synthetic and real-world series, ADAGA's performance is compared against five state-of-the-art algorithms for finding CPs, in terms of F-1 score: BOCPD (Adams & MacKay, 2007), RBOCPDMS (Knoblauch et al., 2018), BINSEG (Scott & Knott, 1974), PELT (Killick et al., 2012), GPTS-CP (Saatçi et al., 2010). The first four algorithms are used with the default initialization provided by their implementations[2]. For the latter algorithm, following Saatçi et al. (2010), we trained the overall model hyperparameters on a portion of the data (the first 30 observations), before running the CP detection algorithm[3]. In line with the protocol used by van den Burg & Williams (2020), from which we obtain the algorithms and the datasets, all algorithms were provided with $\mathbf{t}$ and $\mathbf{y}$, that is, the time indices and the series' observations. Note however that $\mathbf{t}$ is ignored by BOCPD, BINSEG, PELT and RBOCPDMS, mirroring the exact protocol used by the respective authors in their analyses of time series, as done, e.g., by Adams & MacKay (2007). A discussion on the choice of the hyperparameters can be found in the supplementary material. Moreover, following van den

[2]These algorithms are available in the R packages `changepoint` (https://CRAN.R-project.org/package=changepoint) and `ocp` (https://CRAN.R-project.org/package=ocp), and at https://github.com/alan-turing-institute/rbocpdms.

[3]For GPTS-CP, we use the Matlab code accompanying the Doctoral Thesis of Turner (2012).

Burg & Williams (2020), we use a ZERO method, that returns an empty CP set. For BINSEG and PELT, we use two versions of the algorithms, the first aimed at finding CPs in the mean only, the second aimed at finding CPs both in the mean and the variance of the signal. Since GPTS-CP is a GP-based detection scheme, we tested it with different kernels, according to the type of CPs of interest. In particular, the linear kernel was used in the Run Log and Business Inventories datasets, whereas a the sum of an RQ and a constant kernel was used with the rest of the datasets. The minimum window size $|\mathcal{S}|$ for ADAGA is set to 15 points. We run ADAGA with a batch size of 1, to achieve maximum resolution in the location of the CPs. To avoid overly conservative estimates, we set $\delta = 0.6$. An additional comparision with the standard CUSUM detection algorithm (Page, 1954) can be found in the supplementary material. Tables 1 and 2 report F-1 scores, across the datasets, for the aforementioned algorithms. ADAGA is able to outperform its competitors, both on the synthetic and on the real-world datasets. The first interesting result to be noted is the beneficial effect of the GP-based modeling used by ADAGA and GPTS-CP, which both exhibit good results in the datasets with varying linear trends. Conversely, the other algorithms fail at identifying those CPs, and cannot capture the behavior of the signal properly. However, GPTS-CP is heavily influenced by the choice of the prior hyperparameters, which are fixed through the execution of the algorithm. This means that the algorithm struggles to identify those CPs that could be specifically interpreted as sudden changes in the hyperparameters of the underlying GP, such as heteroscedasticity and changes in periodicity, as of Table 1. Furthermore, we can observe that increasing the complexity of the kernel function (e.g., by using an RQ instead of an RBF kernel) does not invalidate the CP detection mechanism used by ADAGA, which gives satisfactory results even in presence of more elaborated kernels.

Detection Delay Besides the F-1 score, another equally important performance metric for online CP detection is the *delay* exhibited by the detection algorithm (Basseville et al., 1993). This quantity describes how many new datapoints an algorithm is required to see after the occurrence of a CP, in order to notify the event. When considering ADAGA, we can observe that its partitioning strategy is based on the GP hyperparameter estimate from the subwindow. Therefore, we need its size, $|\mathcal{S}|$, to be large enough so that maximum likelihood estimation can give meaningful results. Under this assumption, we would expect our algorithm to detect the CP with high probability, once the subwindow includes mainly points coming from the new data-generating process (i.e., after the CP). Thus, our algorithm incurs in a delay that is roughly equal to $|\mathcal{S}|$. This claim is strengthened by the results shown in Table 3. There, we report the average delay (with standard deviation) between the occurrence of a CP in the synthetic time series, and the time at which ADAGA detects it, when the CP is detected (i.e, it is within a margin of 5 points around the true CP). As expected, in the majority of experiments, this delay is smaller than or approximately equal to the subwindow's size, i.e., 15 points. We choose the synthetic time series so as to have multiple noisy realizations. An average delay smaller than the $|\mathcal{S}|$ means that the CP is detected based on just a few samples from the new data-generating process.

5. Conclusion

We have presented ADAGA, an algorithm for scalable GP-based CP detection in time series. We tested our algorithm with two different approximation schemes, namely QFFs and inducing points, and a variety of kernels, showing its ability to model a wide class of CPs, and outperforming state-of-the-art competitors.

Acknowledgements

This research was supported by the Max Planck ETH Center for Learning Systems. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No 815943.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., et al. Tensorflow: A system for large-scale machine learning. OSDI'16, pp. 265–283. USENIX Association, 2016.

Adams, R. P. and MacKay, D. J. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

Basseville, M., Nikiforov, I. V., et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, 1993.

Boucheron, S., Lugosi, G., and Massart, P. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press, 2013.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3): 1–58, 2009.

Chang, W.-C., Li, C.-L., Yang, Y., and Póczos, B. Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*, 2019.

Clifton, L., Clifton, D. A., Pimentel, M. A., Watkinson, P. J., and Tarassenko, L. Gaussian processes for personalized e-health monitoring with wearable sensors. *IEEE*

Transactions on Biomedical Engineering, 60(1):193–197, 2012.

De G. Matthews, A. G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. Gpflow: A Gaussian process library using Tensorflow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

Deisenroth, M. P., Fox, D., and Rasmussen, C. E. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2013.

Galceran, E., Cunningham, A. G., Eustice, R. M., and Olson, E. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots*, 41(6):1367–1382, 2017.

Han, J., Lee, K., Tong, A., and Choi, J. Confirmatory Bayesian online change point detection in the covariance structure of gaussian processes. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2449–2455, 2019.

Hegglin, M. I., Fahey, D. W., McFarland, M., Montzka, S. A., Nash, E. R., et al. *Twenty questions and answers about the ozone layer: 2014 update-scientific assessment of ozone depletion: 2014*. 2015.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 282–290, 2013.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Keshavarz, H., Scott, C., and Nguyen, X. Optimal change point detection in Gaussian processes. *Journal of Statistical Planning and Inference*, 193:151–178, 2018.

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

Knoblauch, J., Jewson, J. E., and Damoulas, T. Doubly robust Bayesian inference for non-stationary streaming data with β -divergences. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Kurt, B., Yıldız, Ç., Ceritli, T. Y., Sankur, B., and Cemgil, A. T. A Bayesian change point model for detecting sip-based ddos attacks. *Digital Signal Processing*, 77:48–62, 2018.

Lavielle, M. and Teyssiere, G. Adaptive detection of multiple change-points in asset price volatility. In *Long Memory in Economics*, pp. 129–156. Springer, 2007.

Lehoucq, R. B., Sorensen, D. C., and Yang, C. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

Lloyd, J., Duvenaud, D., Grosse, R., Tenenbaum, J., and Ghahramani, Z. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

MacKay, D. J. Introduction to Gaussian processes. *NATO ASI Series F: Computer and Systems Sciences*, 168:133–166, 1998.

Masoudnia, S. and Ebrahimpour, R. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2): 275–293, 2014.

Mutny, M. and Krause, A. Efficient high dimensional Bayesian optimization with additivity and quadrature fourier features. In *Advances in Neural Information Processing Systems*, pp. 9005–9016, 2018.

Nguyen-Tuong, D., Peters, J. R., and Seeger, M. Local Gaussian process regression for real time online model learning. In *Advances in Neural Information Processing Systems*, pp. 1193–1200, 2009.

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

Petersen, K. and Pedersen, M. The matrix cookbook, version 20121115. *Technical Univ. Denmark, Kongens Lyngby, Denmark, Tech. Rep*, 3274, 2012.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pp. 1177–1184, 2008.

Saatçi, Y., Turner, R. D., and Rasmussen, C. E. Gaussian process change point models. In *International Conference on Machine Learning*, 2010.

Scott, A. J. and Knott, M. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, pp. 507–512, 1974.

Skates, S. J., Pauler, D. K., and Jacobs, I. J. Screening based on the risk of cancer calculation from Bayesian hierarchical changepoint and mixture models of longitudinal markers. *Journal of the American Statistical Association*, 96(454):429–439, 2001.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pp. 1257–1264, 2006.

Stork, J. A. and Stoyanov, T. Ensemble of sparse gaussian process experts for implicit surface mapping with streaming data. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10758–10764, 2020.

Tartakovsky, A., Nikiforov, I., and Basseville, M. *Sequential analysis: Hypothesis testing and changepoint detection*. CRC Press, 2014.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Turner, R. D. *Gaussian processes for state space models and change point detection*. PhD thesis, University of Cambridge, 2012.

van den Burg, G. J. and Williams, C. K. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*, 2020.

Vehtari, A., Gelman, A., and Gabry, J. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and computing*, 27(5):1413–1432, 2017.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, 17(3):261–272, 2020.

Wainwright, M. J. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.

Williams, C. K. and Seeger, M. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pp. 682–688, 2001.

Woodbury, M. A. *Inverting modified matrices*. Statistical Research Group, 1950.

Table 4. F-1 scores for our CP detection experiments on synthetic data, obtained with a margin of 5 points, and a varying subwindow's size $|\mathcal{S}|$.

$\lvert\mathcal{S}\rvert =$	5	10	40
MEAN SHIFT	0.5 ± 0	0.9 ± 0.2	0.5 ± 0
VARIANCE SHIFT DATA	0.5 ± 0	0.76 ± 0.12	0.5 ± 0
PERIODICITY SHIFT DATA	0.5 ± 0	0.54 ± 0.14	0.5 ± 0

A. Useful Definitions and Lemmas

In this section, we restate one definition and one lemma that are extensively used in our work. Definition A.1 was taken from the work of Wainwright (2019). The lemma can be found in Petersen & Pedersen (2012), Subsection 3.2.2.

Definition A.1 (Subexponential random variable)**.** A centered random variable X is subexponential with non-negative parameters (ν, α) if and only if

$$\mathbb{E}\left[e^{sX}\right] \leq e^{\frac{\nu^2 s^2}{2}},\ \forall |s| < \frac{1}{\alpha}.$$

Lemma A.2 (Woodbury identity– a.k.a. matrix inversion lemma)**.** *Let* $\mathbf{A}$ *and* $\mathbf{C}$ *be invertible matrices. Then, for any matrices* $\mathbf{U}$*,* $\mathbf{V}$ *with proper dimensions, we have*

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}.$$

B. Derivation of the Posterior Covariance Matrix of a GP

Our posterior covariance in Equation (5) of the main paper is not in standard form, but it can readily be retrieved from the standard form found e.g. in (Williams & Rasmussen, 2006):

$$\begin{aligned}
\mathbf{C} - \mathbf{C}(\mathbf{C} + \sigma^2\mathbf{I})^{-1}\mathbf{C} &= (\mathbf{I} - \mathbf{C}(\mathbf{C} + \sigma^2\mathbf{I})^{-1})\mathbf{C} && (12)\\
&= (\mathbf{C} + \sigma^2\mathbf{I} - \mathbf{C})(\mathbf{C} + \sigma^2\mathbf{I})^{-1}\mathbf{C} && (13)\\
&= \sigma^2(\mathbf{C} + \sigma^2\mathbf{I})^{-1}\mathbf{C}. && (14)
\end{aligned}$$

C. Choosing the Subwindow's Size

While we chose the subwindow's size, $|\mathcal{S}|$, to be a hyperparameter, it is also possible to derive it from the experimental setup used. For instance, let us assume that the CPs are given by a sequence of i.i.d. Bernoulli experiments with parameter $p \in (0, 1)$. Then, the distance between two CPs, D, is a geometric random variable, whose cumulative density function is given by

$$\mathbb{P}[D \leq k] = 1 - (1 - p)^k. \tag{15}$$

Now, let $\alpha \in (0, 1)$. Then, we can derive a lower bound on the distance between two CPs that holds with probability $1 - \alpha$:

$$\begin{aligned}
\mathbb{P}[D \leq k] \leq \alpha &\Leftrightarrow 1 - (1 - p)^k \leq \alpha && (16)\\
&\Leftrightarrow (1 - p)^k \geq 1 - \alpha && (17)\\
&\Leftrightarrow k \leq \log_{1-p}(1 - \alpha) && (18)\\
&\Leftrightarrow k \leq \frac{\ln(1 - \alpha)}{\ln(1 - p)} && (19)
\end{aligned}$$

Therefore, setting

$$|\mathcal{S}| = \frac{\ln(1 - \alpha)}{\ln(1 - p)} \tag{20}$$

guarantees that, with probability $1 - \alpha$, $\mathcal{S}$ will not include 2 subsequent CPs. This ensures in turn that each final window will at most contain one CP, for sufficiently well behaved CP-distributions, since the subwindow's size is equal to the minimum window's size.

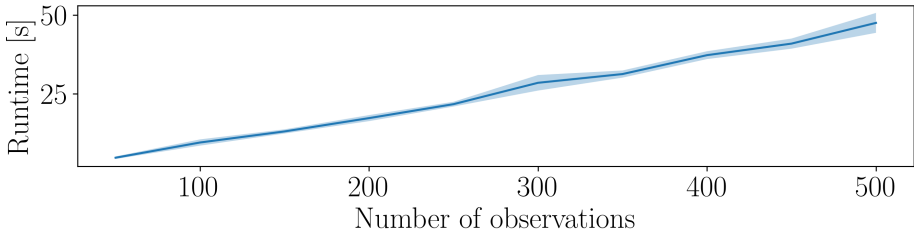


Figure 4. ADAGA's run time is linear w.r.t. an increasing time horizon.

To better appreciate the effects of choosing a wrong $|\mathcal{S}|$, we can consider Table 4. There, we show the F-1 score obtained by processing our synthetic series presented in Section 4 with different subwindow's sizes, an RBF kernel and 10 IPs. As expected from the previous derivations, if $|\mathcal{S}|$ is too small to infer the GP hyperparameters (5 points) or too large, so as to include multiple CPs at the same time (40 points), the performance of ADAGA worsens.

D. Computational Complexity

To calculate the computational complexity of ADAGA, we can proceed as follows. Assume a sequence of n datapoints and a batch size b. Training 2 sparse GPs on the current window of size w_i is known to be $\mathcal{O}(w_i m^2)$. Using a sliding window, the worst-case number of tests being performed is in $\mathcal{O}(n/b)$. As of Proposition 3.8, the complexity of a single test is in $\mathcal{O}(m^3)$, leading to an overall complexity in

$$\mathcal{O}\left(n/b \cdot m^3 + \sum_{i=|\mathcal{S}|}^{n/b} w_i m^2\right) \leq \mathcal{O}(n/b \cdot m^3 + n^2/b \cdot m^2). \tag{21}$$

For a constant upper bound on the window size, e.g., maximum distance between CPs, this simplifies to $\mathcal{O}(n/b)$.

Following these derivations, it can be observed that the *scalability* of ADAGA emerges when considering potentially infinitely long time series. For an approximately fixed maximum window's size (i.e., distance between CPs being detected), ADAGA enjoys a linear runtime. The approximate GP methods presented in Section 2 will only be necessary if these windows grow prohibitively large, giving the practitioner an additional, well-known tool to use in combination with ADAGA. To better understand this, we can consider Figure 4. There, we ran ADAGA (with RBF kernel and IPs) on the time series $y(t) = \sin(0.5t)$ with an increase in mean of 2 units every 25 points, and an increasing horizon. The batch size is set to 10 points, and the mean runtime, along with the standard deviation, are computed over 10 independent noise realizations. As expected, given that the maximum distance between CPs is upper bounded by a constant, our algorithm enjoys a linear runtime. Note that the splits performed by ADAGA are triggered by the detection of CPs, i.e., the maximum window size is not associated to a hardcoded upper bound. An exact GP would be slower (due to the GP trainings being cubic in the size of the current window), but would still enjoy the same linear complexity.

While in our implementation, we used the default parameters provided by `scipy`, who keep it constant, it should be mentioned that the accuracy of Arnoldi's method suffers for increasing number of observations. Thus, it might make sense to increase this number linearly with the amount of observations, which would increase the computational complexity of ADAGA to $\mathcal{O}\left(\sum_{i=|\mathcal{S}|}^{n/b}(w_i + m^3 + w_i m^2)\right)$ However, in our experiments, this was not necessary.

E. Thresholds for the Statistical Test

Here, we provide a detailed proof for Theorems 3.3 and 3.4, which give the thresholds to be used in ADAGA's statistical test. The notation is the same as the one used in Definition 3.1.

E.1. Threshold for Type I Errors

As stated in Subsection 3.4, for the so-called *type I errors*, we are interested in bounding the probability $\mathbb{P}\left[\mathcal{R} \geq \mathcal{T}_I | H_0\right]$, where $\mathcal{T}_I$ is the alternative hypothesis acceptance threshold.

Omitting the conditioning on the null hypothesis, we get the following lemma.

Lemma E.1. *Let* $\mathcal{T}_I = \mathbb{E}[\mathcal{R}|H_0] + t = \mu_{H_0} + t$ *and* $\hat{\mu}_{H_0} = \mathbb{E}\left[\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s\right]$. *Then,*

$$\mathcal{R} \geq \mathcal{T}_I \Leftrightarrow \mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s - \hat{\mu}_{H_0} \leq -t.$$

Proof. By linearity of expectation, we get

$$\begin{aligned}
\mathcal{R} \geq \mathcal{T}_I &\Leftrightarrow \mathcal{R} \geq \mu_{H_0} + t && (22)\\
&\Leftrightarrow -\mathcal{R} \leq -\mu_{H_0} - t && (23)\\
&\Leftrightarrow -\mathcal{R} \leq \mathbb{E}[-\mathcal{R}] - t && (24)\\
&\Leftrightarrow -2\ln\frac{1}{Z_1} + 2\ln Z_{new} + \mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s \\
&\quad \leq \mathbb{E}\left[-2\ln\frac{1}{Z_1} + 2\ln Z_{new} + \mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s\right] - t. && (25)
\end{aligned}$$

Setting

$$c = -2\ln\frac{1}{Z_1} + 2\ln Z_{new}, \tag{26}$$

we have

$$\begin{aligned}
\mathcal{R} \geq \mathcal{T}_I &\Leftrightarrow \mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s + c \leq \mathbb{E}\left[\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s + c\right] - t && (27)\\
&\Leftrightarrow \mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s - \hat{\mu}_{H_0} \leq -t. && (28)
\end{aligned}$$

□

Let us now consider the random variable $\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s$. According to the definition of a subexponential random variable (Definition A.1), we get the following useful characterization of the random variable $\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s - \hat{\mu}_{H_0}$. We will prove the following theorem using similar techniques to (Wainwright, 2019), Example 2.8, and (Han et al., 2019), Theorem 3.1.

Theorem E.2. *Let* $\mathbf{V}_{H_0}$ *be the covariance matrix of the likelihood in the null hypothesis, and let* λ_{i,H_0} *be the* i*-th eigenvalue of the matrix* $\mathbf{V}_{H_0}^{1/2} \mathbf{V}_{new}^{-1} \mathbf{V}_{H_0}^{1/2}$. *The random variable* $\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s - \hat{\mu}_{H_0}$ *is subexponential with parameters* $\left(\sqrt{\sum_i 4\lambda_{i,H_0}^2},\ \max_i \{4\lambda_{i,H_0}\}\right)$.

Proof. In the following, we assume that all the matrices of interest exist. Thus, to ensure their positive definiteness, required since we are computing inverses, we can add a small jitter to their diagonal. We can now observe that the random variable $\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s$ is a linear combination of independent χ^2 variables, which we denote by u_i, with one degree of freedom each. This can be obtained by defining the following two random vectors, with $\mathbf{Q}^T \mathbf{\Lambda} \mathbf{Q}$ being the eigendecomposition of the matrix $\mathbf{V}_{H_0}^{1/2} \mathbf{V}_{new}^{-1} \mathbf{V}_{H_0}^{1/2}$:

$$\mathbf{y}'_s = \mathbf{V}_{H_0}^{-1/2} \mathbf{y}_s \sim \mathcal{N}(\mathbf{y}'_s | \mathbf{0},\ \mathbf{I}), \tag{29}$$

$$\mathbf{y}''_s = \mathbf{Q}\mathbf{y}'_s \sim \mathcal{N}(\mathbf{y}''_s | \mathbf{0},\ \mathbf{I}). \tag{30}$$

Thus, we have

$$\begin{aligned}
\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s &= \mathbf{y}_s^T \mathbf{V}_{H_0}^{-1/2} \mathbf{V}_{H_0}^{1/2} \mathbf{V}_{new}^{-1} \mathbf{V}_{H_0}^{1/2} \mathbf{V}_{H_0}^{-1/2} \mathbf{y}_s && (31)\\
&= \mathbf{y}'^T_s \mathbf{V}_{H_0}^{1/2} \mathbf{V}_{new}^{-1} \mathbf{V}_{H_0}^{1/2} \mathbf{y}'_s && (32)\\
&= \mathbf{y}'^T_s \mathbf{Q}^T \mathbf{\Lambda} \mathbf{Q} \mathbf{y}'_s && (33)\\
&= \mathbf{y}''^T_s \mathbf{\Lambda} \mathbf{y}''_s && (34)\\
&= \sum_i \lambda_{i,H_0} u_i. && (35)
\end{aligned}$$

To lighten the notation, the eigenvalues λ_{i,H_0} of $\mathbf{V}_{H_0}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_0}^{1/2}$ are simply called λ_i, in the rest of the proof. We can now observe that the matrix $\mathbf{V}_{H_0}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_0}^{1/2}$ is positive definite. This can be checked by firstly observing that $\mathbf{V}_{H_0}^{1/2}$ is symmetric, and then applying the definition of positive definite matrices:

$$\mathbf{x}^T\mathbf{V}_{H_0}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_0}^{1/2}\mathbf{x} = \left(\mathbf{V}_{H_0}^{1/2}\mathbf{x}\right)^T \mathbf{V}_{new}^{-1}\left(\mathbf{V}_{H_0}^{1/2}\mathbf{x}\right) \tag{36}$$

$$= \mathbf{q}^T\mathbf{V}_{new}^{-1}\mathbf{q} \tag{37}$$

$$> 0. \tag{38}$$

The last inequality holds since $\mathbf{V}_{new}^{-1}$ is positive definite, as stated at the beginning of this proof. Thus, each λ_i is strictly positive, and therefore $\lambda_i u_i$ follows a Γ distribution with parameters $\left(\frac{1}{2},\ 2\lambda_i\right)$. We know that, if each one of these variables, after centering, is subexponential, then the sum is also subexponential, as reported in (Wainwright, 2019). To see that each centered addendum is subexponential, we can proceed as follows. For $\lambda_i u_i \sim \Gamma(\frac{1}{2}, 2\lambda_i)$, we know that, after centering, the moment generating function is given by

$$\mathbb{E}\left[e^{s(\lambda_i u_i - \mathbb{E}[\lambda_i u_i])}\right] = e^{-\lambda_i s - \frac{1}{2}\ln(1-2\lambda_i s)}. \tag{39}$$

For $s \in \left(0,\ \frac{1}{2\lambda_i}\right)$, using the fact, reported, e.g., in Boucheron et al. (2013), Chapter 2, that

$$-u - \ln(1-u) \leq \frac{u^2}{2(1-u)},\ \forall u \in (0,1), \tag{40}$$

we get

$$\mathbb{E}\left[e^{s(\lambda_i u_i - \mathbb{E}[\lambda_i u_i])}\right] \leq e^{\frac{1}{2}\frac{(2\lambda_i s)^2}{2(1-2\lambda_i s)}}. \tag{41}$$

For $s \in \left(0,\ \frac{1}{4\lambda_i}\right)$, we have that

$$\inf_{s \in \left(0,\ \frac{1}{4\lambda_i}\right)} \{1 - 2\lambda_i s\} = \frac{1}{2}, \tag{42}$$

and so we get

$$\mathbb{E}\left[e^{s(\lambda_i u_i - \mathbb{E}[\lambda_i u_i])}\right] \leq e^{2\lambda_i^2 s^2}. \tag{43}$$

Moreover, we have that the inequality

$$\mathbb{E}\left[e^{s(\lambda_i u_i - \mathbb{E}[\lambda_i u_i])}\right] \leq e^{2\lambda_i^2 s^2} \tag{44}$$

holds also $\forall s \leq 0$. This can be seen by using the well-known logarithmic inequality

$$\ln(1-u) \geq \frac{-u}{1-u},\ \forall u \leq 0, \tag{45}$$

which gives an upper bound on the left hand side. Therefore, it suffices to check that

$$e^{-\lambda_i s - \frac{1}{2}\frac{-2\lambda_i s}{1-2\lambda_i s}} \leq e^{2\lambda_i^2 s^2} \Leftrightarrow -\lambda_i s - \frac{1}{2}\frac{-2\lambda_i s}{1-2\lambda_i s} \leq 2\lambda_i^2 s^2 \tag{46}$$

$$\Leftrightarrow \frac{-\lambda_i s(1-2\lambda_i s) + \lambda_i s - 2\lambda_i^2 s^2(1-2\lambda_i s)}{1-2\lambda_i s} \leq 0 \tag{47}$$

$$\Leftrightarrow \frac{4\lambda_i^3 s^3}{1-2\lambda_i s} \leq 0. \tag{48}$$

The last inequality holds $\forall s \in (-\infty, 0]$ (the numerator is negative, but the denominator is positive over that interval). Therefore,

$$\mathbb{E}\left[e^{s(\lambda_i u_i - \mathbb{E}[\lambda_i u_i])}\right] \leq e^{2\lambda_i^2 s^2} \tag{49}$$

holds $\forall |s| < \frac{1}{4\lambda_i}$, and thus $\lambda_i u_i - \mathbb{E}[\lambda_i u_i]$ is subexponential with parameters $(2\lambda_i,\ 4\lambda_i)$. By linearity of expectation and the aforementioned composition property of subexponential random variables, we get that

$$\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s - \hat{\mu}_{H_0} = \sum_i (\lambda_i u_i - \mathbb{E}[\lambda_i u_i]) \tag{50}$$

is subexponential with parameters $\left(\sqrt{\sum_i 4\lambda_i^2},\ \max_i \{4\lambda_i\}\right)$. □

Subexponential random variables are known to obey to the following tail bound (see, e.g., (Wainwright, 2019), Proposition 2.9).

Theorem E.3. *Let X be a subexponential random variable with parameters* $(\nu,\ \alpha)$*. Then,*

$$\mathbb{P}\left[X \leq -t\right] \leq e^{-\frac{1}{2}\min\left\{\frac{t^2}{\nu^2}, \frac{t}{\alpha}\right\}}.$$

Hence, we can now restate and proof Theorem 3.3, which gives a criterion for choosing a threshold that controls the type I error probability.

Theorem E.4. *Let* $\delta \in (0, 1)$*, and let* λ_{i,H_0}*,* μ_{H_0} *be as before. Setting*

$$\mathcal{T}_I = \mu_{H_0} + \max\left\{\sqrt{8\ln\left(\frac{1}{\delta}\right)\sum_i \lambda_{i,H_0}^2},\ 8\ln\left(\frac{1}{\delta}\right)\max_i\{\lambda_{i,H_0}\}\right\}$$

guarantees a type I error probability of at most δ*.*

Proof. To lighten the notation, λ_{i,H_0} is simply called λ_i, in this proof. Linearity of expectation and Theorem E.3 give

$$\mathbb{P}\left[\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s - \hat{\mu}_{H_0} \leq -t\right] = \mathbb{P}\left[\sum_i (\lambda_i u_i - \mathbb{E}[\lambda_i u_i]) \leq -t\right] \tag{51}$$

$$\leq e^{-\frac{1}{2}\min\left\{\frac{t^2}{\nu^2}, \frac{t}{\alpha}\right\}} \tag{52}$$

$$= e^{-\frac{1}{2}\min\left\{\frac{t^2}{\sum_i 4\lambda_i^2}, \frac{t}{\max_i 4\lambda_i}\right\}}. \tag{53}$$

Such probability is smaller than δ for

$$t \geq \max\left\{\sqrt{2\ln\left(\frac{1}{\delta}\right)\sum_i 4\lambda_i^2},\ 2\ln\left(\frac{1}{\delta}\right)\max_i\{4\lambda_i\}\right\} \tag{54}$$

We can therefore choose $\mathcal{T}_I$ as in the statement of the theorem. □

E.2. Threshold for Type II Errors

Since the subexponential tail bound is symmetric, a similar procedure can be followed to determine the acceptance threshold for controlling the so called *type II errors*. In this case, according to Subsection 3.4, we are interested in bounding the probability $\mathbb{P}\left[\mathcal{R} \leq \mathcal{T}_{II} | H_1\right]$, where $\mathcal{T}_{II}$ is the alternative hypothesis acceptance threshold.

Omitting the conditioning on the alternative hypothesis, we get the following lemma.

Lemma E.5. *Let* $\mathcal{T}_{II} = \mathbb{E}\left[\mathcal{R}|H_1\right] - t = \mu_{H_1} - t$ *and* $\hat{\mu}_{H_1} = \mathbb{E}\left[\mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s\right]$*. Then,*

$$\mathcal{R} \leq \mathcal{T}_{II} \Leftrightarrow \mathbf{y}_s^T \mathbf{V}_{new}^{-1} \mathbf{y}_s - \hat{\mu}_{H_1} \geq +t.$$

Proof. By linearity of expectation, we get

$$\mathcal{R} \leq \mathcal{T}_{II} \Leftrightarrow \mathcal{R} \leq \mu_{H_1} - t \tag{55}$$
$$\Leftrightarrow -\mathcal{R} \geq -\mu_{H_1} + t \tag{56}$$
$$\Leftrightarrow -\mathcal{R} \geq \mathbb{E}[-\mathcal{R}] + t \tag{57}$$
$$\Leftrightarrow -2\ln\frac{1}{Z_1} + 2\ln Z_{new} + \mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s \geq \mathbb{E}\left[-2\ln\frac{1}{Z_1} + 2\ln Z_{new} + \mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s\right] + t. \tag{58}$$

Setting

$$c = -2\ln\frac{1}{Z_1} + 2\ln Z_{new}, \tag{59}$$

we get

$$\Leftrightarrow \mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s + c \geq \mathbb{E}[\mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s + c] + t \tag{60}$$
$$\Leftrightarrow \mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s - \hat{\mu}_{H_1} \geq +t. \tag{61}$$

□

Again, we get a subexponential characterization of the random variable $\mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s - \hat{\mu}_{H_1}$.

Theorem E.6. *Let $\mathbf{V}_{H_1}$ be the covariance matrix of the likelihood in the alternative hypothesis, and let λ_{i,H_1} be the* i*-th eigenvalue of the matrix $\mathbf{V}_{H_1}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_1}^{1/2}$. The random variable $\mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s - \hat{\mu}_{H_1}$ is subexponential with parameters $\left(\sqrt{\sum_i 4\lambda_{i,H_1}^2},\ \max_i\{4\lambda_{i,H_1}\}\right)$.*

Proof. The proof is the same as in Theorem E.2, with $\mathbf{V}_{H_1}$ replacing $\mathbf{V}_{H_0}$. □

Furthermore, we have the following bound for the right tail of a subexponential random variables.

Theorem E.7. *Let X be a subexponential random variable with parameters $(\nu,\ \alpha)$. Then,*

$$\mathbb{P}[X \geq t] \leq e^{-\frac{1}{2}\min\left\{\frac{t^2}{\nu^2}, \frac{t}{\alpha}\right\}}.$$

Hence, we can now restate and proof Theorem 3.4, which gives a criterion for choosing a threshold that controls the type II error probability.

Theorem E.8. *Let $\delta \in (0,1)$, and let λ_{i,H_1}, μ_{H_1} be as before. Setting*

$$\mathcal{T}_{II} = \mu_{H_1} - \max\left\{\sqrt{8\ln\left(\frac{1}{\delta}\right)\sum_i \lambda_{i,H_1}^2},\ 8\ln\left(\frac{1}{\delta}\right)\max_i\{\lambda_{i,H_1}\}\right\}$$

guarantees a type II error probability of at most δ.

Proof. To lighten the notation, λ_{i,H_1} is simply called λ_i, in this proof. We use the same approach as in Theorem 3.3. Linearity of expectation and Theorem E.7 give

$$\mathbb{P}\left[\mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s - \hat{\mu}_{H_1} \geq t\right] = \mathbb{P}\left[\sum_i(\lambda_i u_i - \mathbb{E}[\lambda_i u_i]) \geq t\right] \tag{62}$$
$$\leq e^{-\frac{1}{2}\min\left\{\frac{t^2}{\nu^2}, \frac{t}{\alpha}\right\}} \tag{63}$$
$$= e^{-\frac{1}{2}\min\left\{\frac{t^2}{\sum_i 4\lambda_i^2}, \frac{t}{\max_i 4\lambda_i}\right\}}. \tag{64}$$

We can thus choose the acceptance threshold that leads to a type II error probability smaller than δ as in the statement of the theorem. □

F. Lemmas for an Efficient Implementation of ADAGA

In this section, we restate and proof Lemmas 3.5, 3.6, 3.7, and Proposition 3.8 (in this order), which concern the design of an efficient implementation of ADAGA.

Lemma F.1. *Let $Tr[\cdot]$ be the trace operator, $k \in \{0, 1\}$, and $\mathbf{V}_{new}$ be the covariance of $\mathcal{M}_n$, the expert trained on the subwindow. Then,*

$$\hat{\mu}_{H_k} = Tr\left[\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}\right].$$

Proof. Since $\mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s$ is a scalar, by using linearity of expectation and trace operators we have

$$\begin{aligned}\hat{\mu}_{H_k} &= \mathbb{E}\left[\mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s\right] &(65)\\ &= Tr\left[\mathbb{E}\left(\mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s\right)\right] &(66)\\ &= \mathbb{E}\left[Tr(\mathbf{y}_s^T\mathbf{V}_{new}^{-1}\mathbf{y}_s)\right]. &(67)\end{aligned}$$

By cyclicity of the trace and linearity again, we get

$$\begin{aligned}\hat{\mu}_{H_k} &= \mathbb{E}\left[Tr(\mathbf{y}_s\mathbf{y}_s^T\mathbf{V}_{new}^{-1})\right] &(68)\\ &= Tr\left[\mathbb{E}(\mathbf{y}_s\mathbf{y}_s^T)\mathbf{V}_{new}^{-1}\right]. &(69)\end{aligned}$$

Thus, under the null hypothesis, we get:

$$\hat{\mu}_{H_0} = Tr\left[\mathbf{V}_{H_0}\mathbf{V}_{new}^{-1}\right], \quad (70)$$

and, under the alternative hypothesis,

$$\hat{\mu}_{H_1} = Tr\left[\mathbf{V}_{H_1}\mathbf{V}_{new}^{-1}\right]. \quad (71)$$

□

Lemma F.2. *The eigenvalues of $\mathbf{V}_{H_0}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_0}^{1/2}$ resp. $\mathbf{V}_{H_1}^{1/2}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_1}^{1/2}$ are the same as the ones of $\mathbf{V}_{H_0}\mathbf{V}_{new}^{-1}$ resp. $\mathbf{V}_{H_1}\mathbf{V}_{new}^{-1}$.*

Proof. This well-known linear-algebraic result can be found, e.g., in Petersen & Pedersen (2012), Subsection 5.2.3. □

Lemma F.3. *Let $\boldsymbol{\lambda}_{H_0}$ and $\boldsymbol{\lambda}_{H_1}$ be the vectors of the eigenvalues of $\mathbf{V}_{H_0}\mathbf{V}_{new}^{-1}$ resp. $\mathbf{V}_{H_1}\mathbf{V}_{new}^{-1}$. Moreover, let $k \in \{0, 1\}$. Then,*

$$\sum_i \lambda_{i,H_k}^2 = ||\boldsymbol{\lambda}_{H_k}||_2^2 = Tr\left[\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}\right].$$

Proof. Let $\mathbf{Q}^T\boldsymbol{\Lambda}\mathbf{Q}$ be the eigendecomposition of $\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}$. Then, orthogonality of $\mathbf{Q}$ and cyclicity of trace give

$$\begin{aligned}Tr\left[\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}\mathbf{V}_{H_k}\mathbf{V}_{new}^{-1}\right] &= Tr\left[\mathbf{Q}^T\boldsymbol{\Lambda}\mathbf{Q}\mathbf{Q}^T\boldsymbol{\Lambda}\mathbf{Q}\right] &(72)\\ &= Tr\left[\boldsymbol{\Lambda}^2\right] &(73)\\ &= \sum_i \lambda_{i,H_k}^2. &(74)\end{aligned}$$

□

Proposition F.4. *Let m be the number of inducing points (or quadrature Fourier features) used. Matrices $\mathbf{V}_{H_0}$, $\mathbf{V}_{H_1}$ and $\mathbf{V}_{new}^{-1}$ can be computed in $\mathcal{O}(m^3)$.*

Proof. In the proof, we work with the inducing points, as described in Section 2. If the QFFs are used, the same substitutions described in the aforementioned subsection apply. Firstly, we denote by $\mathbf{K}$ any prior covariance matrices whose hyperparameters were learned from the whole current window, and $\mathbf{H}$ those whose hyperparameters were learned on the subwindow only. Moreover, let σ^2 denote the noise variance learned from the whole window, and ξ^2 the one learned from the subwindow only. Since

$$\mathbf{V}_{H_0} = \mathbf{K}_{n,m}\mathbf{K}_{m,m}^{-1}\mathbf{K}_{m,n} + \sigma^2\mathbf{I}, \quad (75)$$

this matrix can be computed in $\mathcal{O}(m^3)$. Moreover, since

$$\mathbf{V}_{new} = \mathbf{H}_{n,m}\mathbf{H}_{m,m}^{-1}\mathbf{H}_{m,n} + \xi^2\mathbf{I}, \tag{76}$$

its inverse can also be computed efficiently by directly applying Lemma A.2. The same holds for the matrix

$$\mathbf{V}_{H_1} = (\mathbf{V}_{H_0}^{-1} + \mathbf{V}_{new}^{-1})^{-1}, \tag{77}$$

as we will show in the following. By matrix inversion lemma, we have

$$\mathbf{V}_{H_1} = \left[(\mathbf{K}_{n,m}\mathbf{K}_{m,m}^{-1}\mathbf{K}_{m,n} + \sigma^2\mathbf{I})^{-1} + (\mathbf{H}_{n,m}\mathbf{H}_{m,m}^{-1}\mathbf{H}_{m,n} + \xi^2\mathbf{I})^{-1}\right]^{-1} \tag{78}$$

$$\begin{aligned} &= \left[\frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^2}\mathbf{K}_{n,m}(\sigma^2\mathbf{K}_{m,m} + \mathbf{K}_{m,n}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\right. \\ &\quad \left. + \frac{1}{\xi^2}\mathbf{I} - \frac{1}{\xi^2}\mathbf{H}_{n,m}(\xi^2\mathbf{H}_{m,m} + \mathbf{H}_{m,n}\mathbf{H}_{n,m})^{-1}\mathbf{H}_{m,n}\right]^{-1} \end{aligned} \tag{79}$$

$$\begin{aligned} &= \left\{\left[\frac{\sigma^2+\xi^2}{\sigma^2\xi^2}\mathbf{I} - \frac{1}{\sigma^2}\mathbf{K}_{n,m}(\sigma^2\mathbf{K}_{m,m} + \mathbf{K}_{m,n}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\right]\right. \\ &\qquad \left. - \frac{1}{\xi^2}\mathbf{H}_{n,m}(\xi^2\mathbf{H}_{m,m} + \mathbf{H}_{m,n}\mathbf{H}_{n,m})^{-1}\mathbf{H}_{m,n}\right\}^{-1}. \end{aligned} \tag{80}$$

Now, let

$$\mathbf{A} = \left[\frac{\sigma^2+\xi^2}{\sigma^2\xi^2}\mathbf{I} - \frac{1}{\sigma^2}\mathbf{K}_{n,m}(\sigma^2\mathbf{K}_{m,m} + \mathbf{K}_{m,n}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\right]. \tag{81}$$

We have

$$\mathbf{V}_{H_1} = \left\{\mathbf{A} - \frac{1}{\xi^2}\mathbf{H}_{n,m}(\xi^2\mathbf{H}_{m,m} + \mathbf{H}_{m,n}\mathbf{H}_{n,m})^{-1}\mathbf{H}_{m,n}\right\}^{-1} \tag{82}$$

$$\begin{aligned} &= \mathbf{A}^{-1} + \frac{1}{\xi^2}\mathbf{A}^{-1}\mathbf{H}_{n,m} \\ &\quad \cdot \left(\xi^2\mathbf{H}_{m,m} + \mathbf{H}_{m,n}\mathbf{H}_{n,m} - \frac{1}{\xi^2}\mathbf{H}_{m,n}\mathbf{A}^{-1}\mathbf{H}_{n,m}\right)^{-1}\mathbf{H}_{m,n}\mathbf{A}^{-1}. \end{aligned} \tag{83}$$

Thus, $\mathbf{V}_{H_1}$ can be computed in $\mathcal{O}(m^3)$, provided that $\mathbf{A}$ can be computed efficiently. This is also true, by matrix inversion lemma:

$$\mathbf{A}^{-1} = \left[\left(\frac{\sigma^2+\xi^2}{\sigma^2\xi^2}\right)\mathbf{I} - \frac{1}{\sigma^2}\mathbf{K}_{n,m}(\sigma^2\mathbf{K}_{m,m} + \mathbf{K}_{m,n}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\right]^{-1} \tag{84}$$

$$= \left[\alpha\mathbf{I} - \frac{1}{\sigma^2}\mathbf{K}_{n,m}(\sigma^2\mathbf{K}_{m,m} + \mathbf{K}_{m,n}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\right]^{-1} \tag{85}$$

$$= \frac{1}{\alpha}\left[\mathbf{I} + \frac{1}{\sigma^2}\mathbf{K}_{n,m}\left(\alpha\sigma^2\mathbf{K}_{m,m} + \alpha\mathbf{K}_{m,n}\mathbf{K}_{n,m} - \frac{1}{\sigma^2}\mathbf{K}_{m,n}\mathbf{K}_{n,m}\right)^{-1}\mathbf{K}_{m,n}\right]. \tag{86}$$

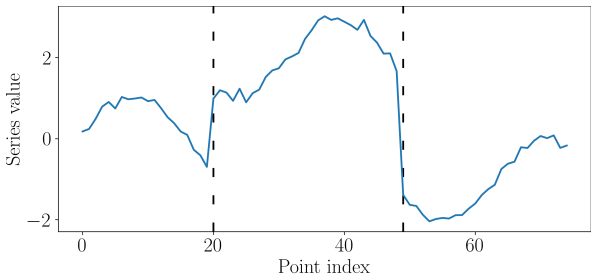
□

G. Extended CP Detection Experiments

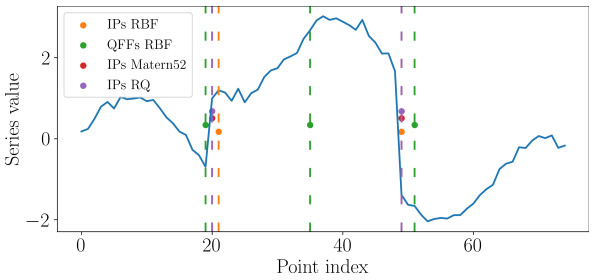
In this section, we discuss the CP detection experiments of Section 4 in detail, reporting additional plots and tables, along with the description of the datasets used. Furthermore we show the location of the CPs detected by our algorithms.

G.1. Description of the Datasets and Annotations

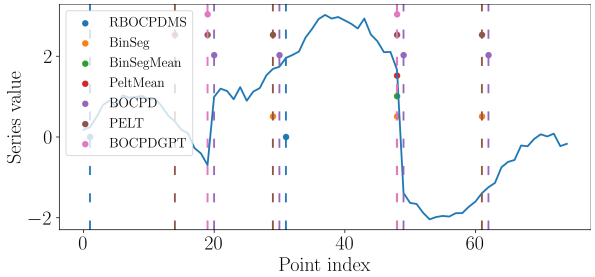
In this subsection we describe the datasets used, and we show plots containing the ground truth and the CP locations returned in our benchmarking. The first three datasets are synthetic. In the plots, we show only the first noisy realization



(a) Annotated

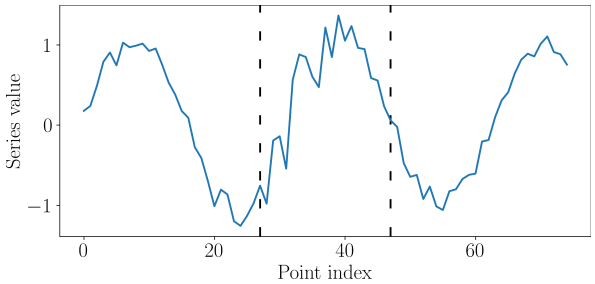


(b) ADAGA

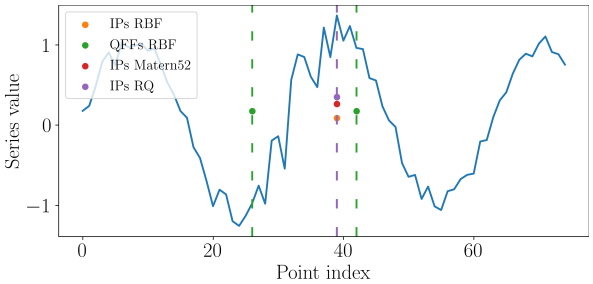


(c) Benchmarks

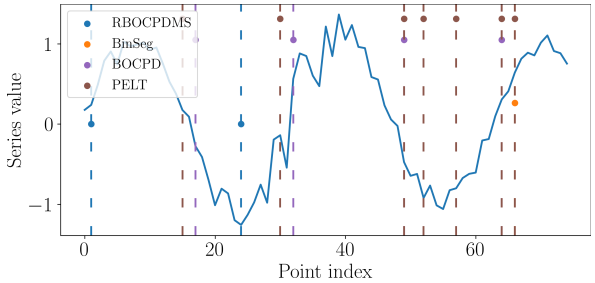
Figure 5. Series with CPs in the mean of the signal. Figure 5a shows the ground truth locations. Vertical lines correspond to the locations. Figure 5b shows the locations detected by ADAGA, Figure 5c the ones computed by the competitors.



(a) Annotated

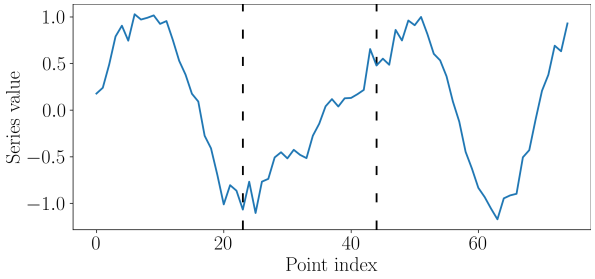


(b) ADAGA

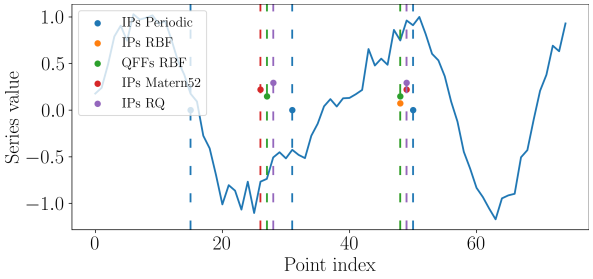


(c) Benchmarks

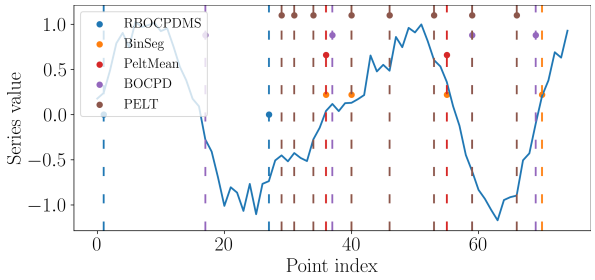
Figure 6. Series with CPs in the noise variance. Figure 6a shows the ground truth locations. Vertical lines correspond to the locations. Figure 6b shows the locations detected by ADAGA, Figure 6c the ones computed by the competitors.



(a) Annotated

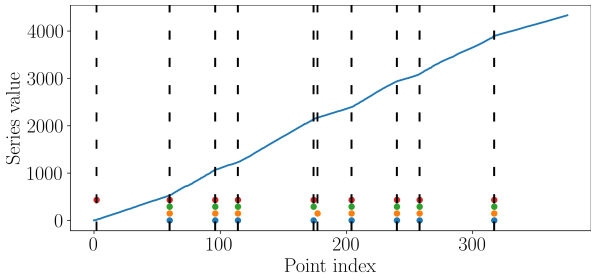


(b) ADAGA

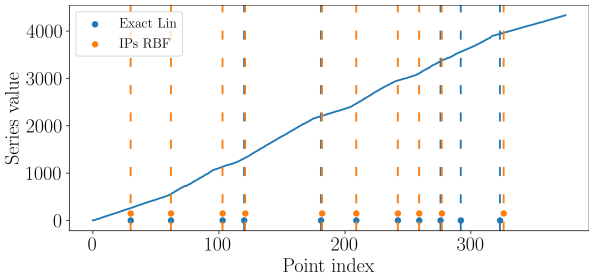


(c) Benchmarks

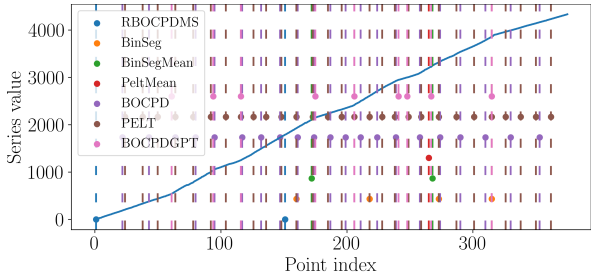
Figure 7. Series with CPs in the periodicity of the signal. Figure 7a shows the ground truth locations. Vertical lines correspond to the locations. Figure 7b shows the locations detected by ADAGA, Figure 7c the ones computed by the competitors.



(a) Annotated

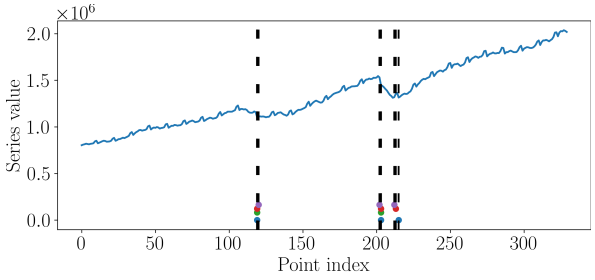


(b) ADAGA

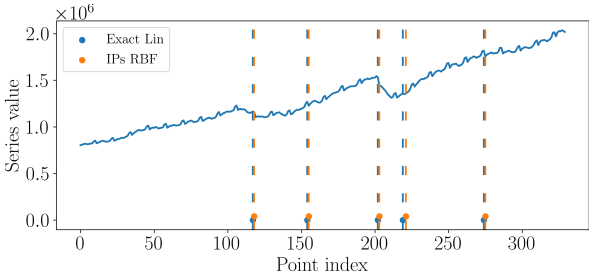


(c) Benchmarks

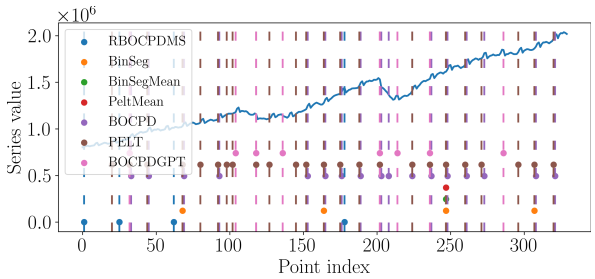
Figure 8. Run Log series. Figure 8a shows the ground truth locations. Vertical lines correspond to the locations. Figure 8b shows the locations detected by ADAGA, Figure 8c the ones computed by the competitors.



(a) Annotated

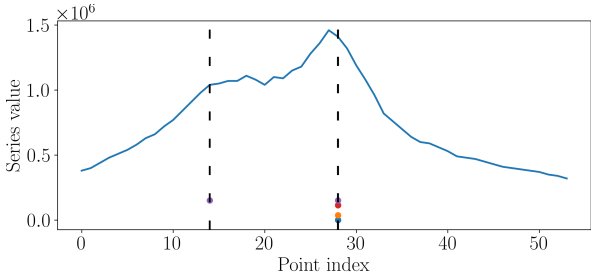


(b) ADAGA

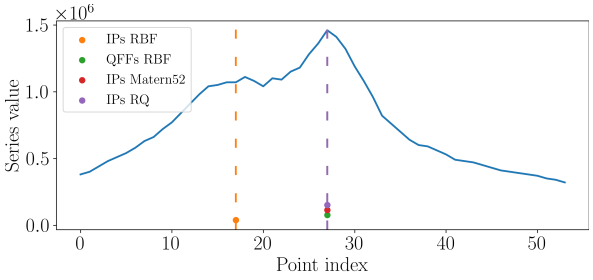


(c) Benchmarks

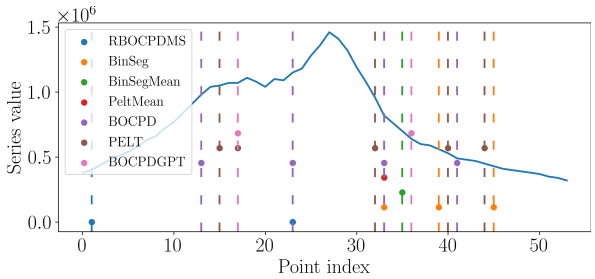
Figure 9. Business Inventories series. Figure 9a shows the ground truth locations. Vertical lines correspond to the locations. Figure 9a shows the ground truth locations. Vertical lines correspond to the locations. Figure 9b shows the locations detected by ADAGA, Figure9c the ones computed by the competitors.



(a) Annotated

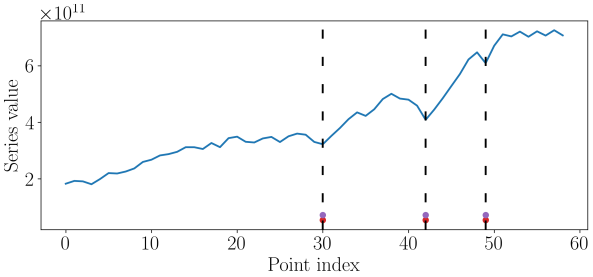


(b) ADAGA

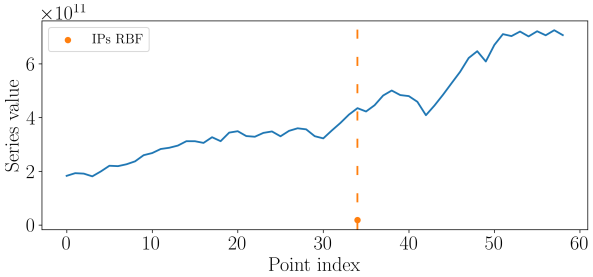


(c) Benchmarks

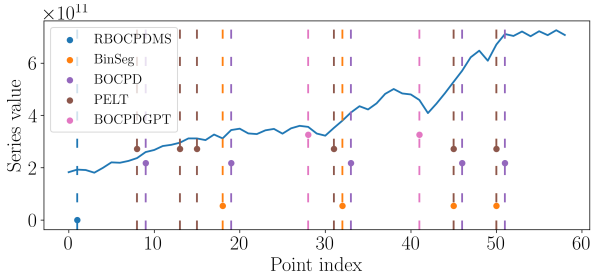
Figure 10. Ozone series. Figure 10a shows the ground truth locations. Vertical lines correspond to the locations. Figure 10b shows the locations detected by ADAGA, Figure 10c the ones computed by the competitors.



(a) Annotated

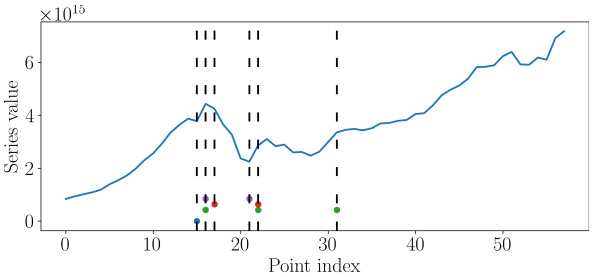


(b) ADAGA

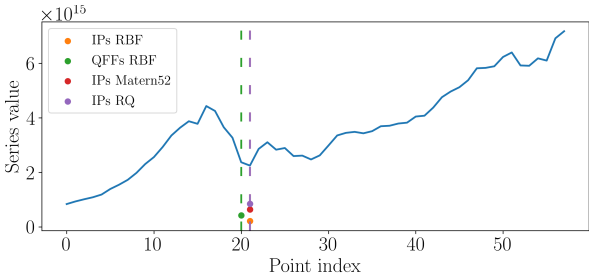


(c) Benchmarks

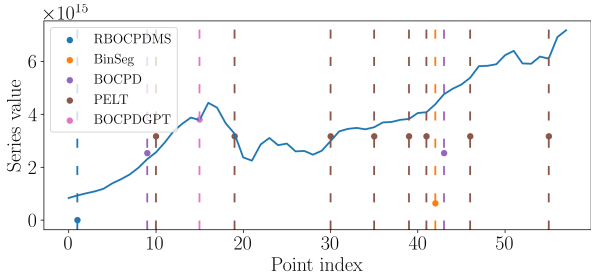
Figure 11. Argentina's GDP series. Figure 11a shows the ground truth locations. Vertical lines correspond to the locations. Figure 11b shows the locations detected by ADAGA, Figure 11c the ones computed by the competitors.



(a) Annotated

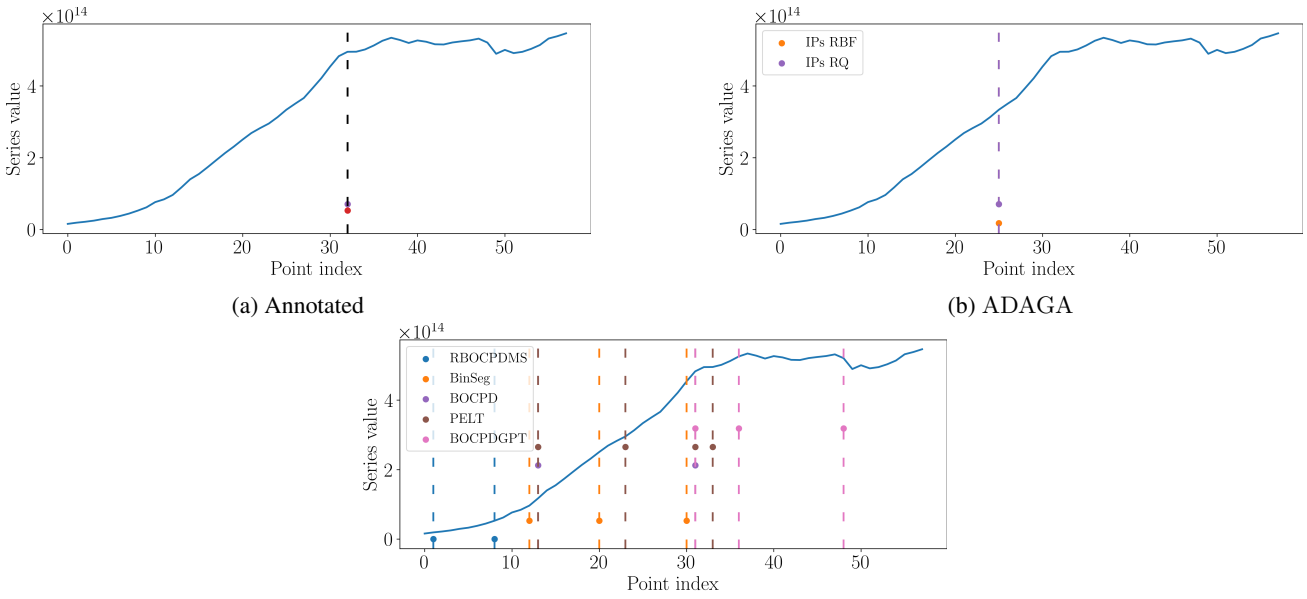


(b) ADAGA



(c) Benchmarks

Figure 12. Iran's GDP series. Figure 12a shows the ground truth locations. Vertical lines correspond to the locations. Figure 12b shows the locations detected by ADAGA, Figure 12c the ones computed by the competitors.



(a) Annotated

(b) ADAGA

(c) Benchmarks

Figure 13. Japan's GDP series. Figure 13a shows the ground truth locations. Vertical lines correspond to the locations. Figure 13b shows the locations detected by ADAGA, Figure 13c the ones computed by the competitors.

Table 5. Average precision scores (with standard deviation) of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across three synthetic datasets (10 noisy realizations per dataset). A margin of 5 points was used. Bold values indicate the highest average score for the dataset. Last column shows the average score and standard deviation across all noisy realizations.

ALGORITHM	MEAN SHIFT DATA	VARIANCE SHIFT DATA	PERIODICITY SHIFT DATA	AVERAGE
ADAGA (QFFs, RBF)	0.61 ± 0.11	0.72 ± 0.26	0.58 ± 0.17	0.64 ± 0.2
ADAGA (IPs, RBF)	$\mathbf{1.0} \pm 0$	0.75 ± 0.25	0.95 ± 0.15	$\mathbf{0.9} \pm 0.2$
ADAGA (IPs, Matern52)	$\mathbf{1.0} \pm 0$	0.75 ± 0.25	0.9 ± 0.2	0.88 ± 0.21
ADAGA (IPs, RQ)	$\mathbf{1.0} \pm 0$	0.75 ± 0.25	0.95 ± 0.15	$\mathbf{0.9} \pm 0.2$
ADAGA (IPs, periodic)	–	–	0.25 ± 0	0.25 ± 0
BINSEG (mean)	0.67 ± 0	$\mathbf{1.0} \pm 0$	0.53 ± 0.16	0.73 ± 0.22
BINSEG (mean & var)	0.5 ± 0	0.32 ± 0.03	0.27 ± 0.074	0.36 ± 0.11
PELT (mean)	0.67 ± 0	0.89 ± 0.22	0.37 ± 0.09	0.64 ± 0.26
PELT (mean & var)	0.37 ± 0.08	0.39 ± 0.1	0.27 ± 0.07	0.34 ± 0.1
BOCPD	0.57 ± 0.06	0.52 ± 0.1	0.37 ± 0.09	0.49 ± 0.12
RBOCPDMS	0.29 ± 0.04	0.52 ± 0.05	0.56 ± 0.12	0.46 ± 0.14
GPTS-CP (RQ+const)	0.91 ± 0.21	0.8 ± 0.26	$\mathbf{1.0} \pm 0$	$\mathbf{0.9} \pm 0.21$

Table 6. Precision scores of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across six real-world datasets. A margin of 5 points was used. Bold values indicate the highest score for the dataset. Last column shows the average score across the datasets.

ALGORITHM	RUN LOG	BUSINV	OZONE	GDP IRAN	GDP ARGENTINA	GDP JAPAN	AVERAGE
ADAGA (exact, linear)	0.5	**0.67**	–	–	–	–	0.59
ADAGA (IPs, linear)	0.55	0.5	–	–	–	–	0.53
ADAGA (QFFs, RBF)	–	–	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
ADAGA (IPs, RBF)	–	–	**1.0**	**1.0**	**1.0**	0.5	0.88
ADAGA (IPs, Matern52)	–	–	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
ADAGA (IPs, RQ)	–	–	**1.0**	**1.0**	**1.0**	0.5	0.88
BINSEG (mean)	0.5	0.33	0.67	0.5	**1.0**	0.5	0.58
BINSEG (mean & var)	0.33	0.17	0.4	0.33	0.67	0.4	0.38
PELT (mean)	0.33	0.33	**1.0**	0.5	**1.0**	0.5	0.61
PELT (mean & var)	0.29	0.12	0.43	0.3	0.5	0.33	0.33
BOCPD	0.36	0.17	0.6	0.33	0.67	0.67	0.47
RBOCPDMS	0.67	0.2	0.67	0.5	0.5	0.33	0.48
GPTS-CP (linear+const)	**0.8**	0.44	–	–	–	–	0.62
GPTS-CP (RQ+const)	–	–	0.67	**1.0**	**1.0**	0.5	0.79

Table 7. Average recall scores of ADAGA for CP detection (with standard deviation), implemented both with QFFs and inducing points (IPs), and six comparable algorithms across three synthetic datasets (10 noisy realizations per dataset). A margin of 5 points was used. Bold values indicate the highest average score for the dataset. Last column shows the average score and standard deviation across all noisy realizations.

ALGORITHM	MEAN SHIFT DATA	VARIANCE SHIFT DATA	PERIODICITY SHIFT DATA	AVERAGE
ADAGA (QFFs, RBF)	0.77 ± 0.15	0.8 ± 0.22	0.5 ± 0.22	0.69 ± 0.24
ADAGA (IPs, RBF)	$\mathbf{1.0} \pm 0$	0.5 ± 0.17	0.43 ± 0.15	0.64 ± 0.28
ADAGA (IPs, Matern52)	$\mathbf{1.0} \pm 0$	0.5 ± 0.17	0.47 ± 0.22	0.66 ± 0.29
ADAGA (IPs, RQ)	$\mathbf{1.0} \pm 0$	0.5 ± 0.17	0.47 ± 0.22	0.67 ± 0.29
ADAGA (IPs, periodic)	–	–	0.33 ± 0	0.33 ± 0
BINSEG (mean)	0.67 ± 0	0.33 ± 0	0.33 ± 0	0.44 ± 0.16
BINSEG (mean & var)	$\mathbf{1.0} \pm 0$	0.33 ± 0	0.53 ± 0.16	0.62 ± 0.29
PELT (mean)	0.67 ± 0	0.4 ± 0.2	0.33 ± 0	0.47 ± 0.18
PELT (mean & var)	$\mathbf{1.0} \pm 0$	$\mathbf{0.93} \pm 0.13$	$\mathbf{0.67} \pm 0.15$	$\mathbf{0.87} \pm 0.18$
BOCPD	0.93 ± 0.13	0.87 ± 0.16	0.63 ± 0.18	0.81 ± 0.21
RBOCPDMS	0.33 ± 0	0.37 ± 0.1	0.5 ± 0.22	0.4 ± 0.16
GPTS-CP (RQ+const)	$\mathbf{1.0} \pm 0$	0.53 ± 0.22	0.33 ± 0	0.62 ± 0.31
ZERO	0.33 ± 0	0.33 ± 0	0.33 ± 0	0.33 ± 0

Table 8. Recall scores of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across six real-world datasets. A margin of 5 points was used. Bold values indicate the highest score for the dataset. Last column shows the average score across the datasets.

ALGORITHM	RUN LOG	BUSINV	OZONE	GDP IRAN	GDP ARGENTINA	GDP JAPAN	AVERAGE
ADAGA (exact, linear)	0.66	0.9	–	–	–	–	0.78
ADAGA (IPs, linear)	0.66	0.85	–	–	–	–	0.76
ADAGA (QFFs, RBF)	–	–	0.93	0.77	0.7	0.8	0.8
ADAGA (IPs, RBF)	–	–	0.63	0.67	0.8	0.8	0.73
ADAGA (IPs, Matern52)	–	–	0.93	0.67	0.7	0.8	0.78
ADAGA (IPs, RQ)	–	–	0.93	0.67	0.7	0.8	0.78
BINSEG (mean)	0.37	0.42	0.63	0.48	0.8	0.8	0.57
BINSEG (mean & var)	0.37	0.42	0.93	0.48	**1.0**	**1.0**	0.70
PELT (mean)	0.29	0.42	**1.0**	0.48	0.8	0.8	0.63
PELT (mean & var)	**0.98**	0.85	**1.0**	**0.82**	**1.0**	**1.0**	0.94
BOCPD	0.89	0.73	**1.0**	0.48	**1.0**	**1.0**	0.85
RBOCPDMS	0.31	0.42	0.93	0.48	0.7	0.8	0.61
GPTS-CP (linear+const)	0.89	**1.0**	–	–	–	–	**0.95**
GPTS-CP (RQ+const)	–	–	0.63	0.77	0.9	**1.0**	0.83
ZERO	0.29	0.42	0.57	0.48	0.7	0.8	0.54

Table 9. Average F-1 scores (with standard deviation) of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across three synthetic datasets (10 noisy realizations per dataset). A margin of 0 points was used. Bold values indicate the highest average score for the dataset. Last column shows the average score and standard deviation across all noisy realizations. As expected, since the annotations are arbitrary within a margin around the true CPs, all the algorithms are outperformed by the ZERO method, which results in an empty CP set. A larger margin is needed to get meaningful results.

ALGORITHM	MEAN SHIFT DATA	VARIANCE SHIFT DATA	PERIODICITY SHIFT DATA	AVERAGE
ADAGA (QFFs, RBF)	0.3 ± 0.02	0.34 ± 0.11	0.37 ± 0.03	0.34 ± 0.07
ADAGA (IPs, RBF)	0.93 ± 0.13	0.4 ± 0	0.46 ± 0.05	0.6 ± 0.25
ADAGA (IPs, Matern52)	$\mathbf{1.0} \pm 0$	0.4 ± 0	0.44 ± 0.06	0.61 ± 0.28
ADAGA (IPs, RQ)	$\mathbf{1.0} \pm 0$	0.4 ± 0	0.45 ± 0.06	$\mathbf{0.62} \pm 0.27$
ADAGA (IPs, periodic)	–	–	0.29 ± 0	0.29 ± 0
BINSEG (mean)	0.33 ± 0	$\mathbf{0.5} \pm 0$	0.4 ± 0.04	0.41 ± 0.07
BINSEG (mean & var)	0.22 ± 0	0.32 ± 0.02	0.23 ± 0.01	0.26 ± 0.05
PELT (mean)	0.33 ± 0	0.46 ± 0.09	0.34 ± 0.04	0.38 ± 0.08
PELT (mean & var)	0.18 ± 0.03	0.27 ± 0.1	0.21 ± 0.03	0.22 ± 0.07
BOCPD	0.71 ± 0.09	0.28 ± 0.08	0.25 ± 0.01	0.41 ± 0.22
RBOCPDMS	0.31 ± 0.02	0.39 ± 0.02	0.36 ± 0.04	0.35 ± 0.05
GPTS-CP (RQ+const)	0.93 ± 0.15	0.43 ± 0.09	$\mathbf{0.5} \pm 0$	$\mathbf{0.62} \pm 0.25$
ZERO	0.5 ± 0	$\mathbf{0.5} \pm 0$	$\mathbf{0.5} \pm 0$	0.5 ± 0

Table 10. F-1 scores of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across six real-world datasets. A margin of 0 points was used. Bold values indicate the highest score for the dataset. Last column shows the average score across the datasets. As expected, since the annotations are arbitrary within a margin around the true CPs, all the algorithms are outperformed by the ZERO method, which results in an empty CP set. A larger margin is needed to get meaningful results.

ALGORITHM	RUN LOG	BUSINV	OZONE	GDP IRAN	GDP ARGENTINA	GDP JAPAN	AVERAGE
ADAGA (exact, linear)	0.12	0.39	–	–	–	–	0.26
ADAGA (IPs, linear)	0.14	0.42	–	–	–	–	0.28
ADAGA (QFFs, RBF)	–	–	0.53	0.49	0.82	0.89	0.68
ADAGA (IPs, RBF)	–	–	0.53	0.71	0.58	0.62	0.61
ADAGA (IPs, Matern52)	–	–	0.53	0.71	**0.82**	**0.89**	**0.74**
ADAGA (IPs, RQ)	–	–	0.53	0.71	**0.82**	0.62	0.67
BINSEG (mean)	0.27	0.37	0.42	0.49	0.58	0.62	0.46
BINSEG (mean & var)	0.21	0.24	0.30	0.39	0.27	0.32	0.29
PELT (mean)	0.31	0.37	0.42	0.49	0.58	0.62	0.46
PELT (mean & var)	0.06	0.13	0.23	0.17	0.21	0.28	0.18
BOCPD	0.21	0.19	0.30	0.39	0.27	0.47	0.30
RBOCPDMS	0.31	0.27	0.42	0.49	0.58	0.47	0.42
GPTS-CP (linear+const)	0.15	0.57	–	–	–	–	0.36
GPTS-CP (RQ+const)	–	–	0.42	**0.75**	0.73	0.67	0.64
ZERO	**0.45**	**0.59**	**0.72**	0.65	**0.82**	**0.89**	0.69

being processed (out of 10), for illustrative purposes. The remaining datasets are presented by van den Burg & Williams (2020). Omitted algorithms return an empty CP set for the dataset considered. BOCPDGPT is an alias for GPTS-CP, PELT stands for the mean and variance version of PELT, and BinSeg stands for the mean and variance version of binary segmentation. IPs stands for ADAGA implemented with inducing points, QFFs stands for ADAGA implemented with QFFs.

CPs in Mean This series contains a signal with two CPs in mean. These were obtained by summing the function

$$x_1(t) = \sin(0.5t), \tag{87}$$

and a constant offset of 0 (for the first 20 observations), 2 (between observations 20 and 49) and -1 (for the remaining 26 observations). In all segments, a zero-mean, additive Gaussian noise with standard deviation of 10^{-1} was used. Figure 5 shows the locations of these CPs, along with the ones computed in our benchmark study. Vertical dashed lines correspond to such locations.

CPs in Variance This series contains a signal with two CPs in noise variance. These were obtained by corrupting the function

$$x_1(t) = \sin(0.5t), \tag{88}$$

with a zero-mean, additive Gaussian noise with standard deviation of 10^{-1} (for the first 23 observations), $3 \cdot 10^{-1}$ (between observations 23 and 44) and $0.8 \cdot 10^{-1}$ (for the remaining 31 observations). Figure 6 shows the locations of these CPs, along with the ones retrieved in our benchmark study. Vertical dashed lines correspond to such locations.

Table 11. Average F-1 scores (with standard deviation) of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across three synthetic datasets (10 noisy realizations per dataset). A margin of 10 points was used. Bold values indicate the highest average score for the dataset. Last column shows the average score and standard deviation across all noisy realizations.

ALGORITHM	MEAN SHIFT DATA	VARIANCE SHIFT DATA	PERIODICITY SHIFT DATA	AVERAGE
ADAGA (QFFs, RBF)	0.82 ± 0.08	$\mathbf{0.93} \pm 0.07$	$\mathbf{0.87} \pm 0.12$	$\mathbf{0.87} \pm 0.1$
ADAGA (IPs, RBF)	$\mathbf{1.0} \pm 0$	0.8 ± 0	0.62 ± 0.15	0.81 ± 0.18
ADAGA (IPs, Matern52)	$\mathbf{1.0} \pm 0$	0.8 ± 0	0.63 ± 0.19	0.81 ± 0.19
ADAGA (IPs, RQ)	$\mathbf{1.0} \pm 0$	0.8 ± 0	0.64 ± 0	0.81 ± 0.18
ADAGA (IPs, periodic)	–	–	0.86 ± 0	0.86 ± 0
BINSEG (mean)	$\mathbf{1.0} \pm 0$	0.5 ± 0	0.79 ± 0.11	0.76 ± 0.22
BINSEG (mean & var)	0.67 ± 0	0.32 ± 0.02	0.68 ± 0.03	0.56 ± 0.17
PELT (mean)	$\mathbf{1.0} \pm 0$	0.51 ± 0.09	0.84 ± 0.11	0.78 ± 0.22
PELT (mean & var)	0.53 ± 0.08	0.59 ± 0.08	0.58 ± 0.09	0.57 ± 0.09
BOCPD	0.73 ± 0.05	0.75 ± 0	0.74 ± 0.02	0.74 ± 0.03
RBOCPDMS	0.31 ± 0.02	0.43 ± 0.08	0.55 ± 0.16	0.43 ± 0.14
GPTS-CP (RQ+const)	0.93 ± 0.15	0.69 ± 0.2	0.5 ± 0	0.43 ± 0.23
ZERO	0.5 ± 0	$\mathbf{0.5} \pm 0$	0.5 ± 0	0.5 ± 0

Table 12. F-1 scores of ADAGA for CP detection, implemented both with QFFs and inducing points (IPs), and six comparable algorithms across six real-world datasets. A margin of 10 points was used. Bold values indicate the highest score for the dataset. Last column shows the average score across the datasets.

ALGORITHM	RUN LOG	BUSINV	OZONE	GDP IRAN	GDP ARGENTINA	GDP JAPAN	AVERAGE
ADAGA (exact, linear)	0.85	**0.8**	–	–	–	–	0.82
ADAGA (IPs, linear)	0.89	**0.8**	–	–	–	–	0.85
ADAGA (QFFs, RBF)	–	–	0.97	**0.87**	0.82	0.89	0.89
ADAGA (IPs, RBF)	–	–	0.78	**0.87**	0.89	**1.0**	0.89
ADAGA (IPs, Matern52)	–	–	0.97	**0.87**	0.82	0.89	0.89
ADAGA (IPs, RQ)	–	–	0.97	**0.87**	0.82	**1.0**	**0.91**
BINSEG (mean)	0.71	0.37	**1.0**	0.49	0.89	**1.0**	0.74
BINSEG (mean & var)	0.35	0.24	0.75	0.71	0.8	0.57	0.57
PELT (mean)	0.48	0.37	**1.0**	0.49	0.89	**1.0**	0.70
PELT (mean & var)	0.52	0.32	0.60	0.67	0.67	0.50	0.55
BOCPD	0.62	0.34	0.75	0.71	0.80	0.80	0.67
RBOCPDMS	0.42	0.27	0.78	0.49	0.58	0.47	0.5
GPTS-CP (linear+const)	**0.94**	0.62	–	–	–	–	0.78
GPTS-CP (RQ+const)	–	–	**1.0**	**0.87**	**0.95**	0.67	0.87
ZERO	0.45	0.59	0.72	0.65	0.82	0.89	0.69

CPs in Periodicity This series contains a signal with two CPs in periodicity. These were obtained by concatenating the functions

$$x_1(t) = \sin(0.5t), \tag{89}$$

for the first 27 observations,

$$x_2(t) = \sin(0.2t), \tag{90}$$

between observations 27 and 47,

$$x_3(t) = \sin(0.6t), \tag{91}$$

for the remaining 28 observations. In all segments, a zero-mean, additive Gaussian noise with standard deviation of 10^{-1} was used. Figure 7 shows the locations of these CPs, along with the ones retrieved in our benchmark study. Vertical dashed lines correspond to such locations.

Run Log This dataset contains the second series of the Run Log dataset introduced by van den Burg & Williams (2020). This consists of the cumulative distance traveled by a runner who alternates between walking and running. Figure 8 shows the annotated and the computed plots.

Business Inventories This dataset contains the monthly number of business inventories, in USD, obtained from the US Census Bureau. Figure 9 shows the annotated and the computed plots.

Ozone This dataset contains the levels of ozone depleting substances, obtained from `www.ourworldindata.org`, and originally shown by Hegglin et al. (2015). Figure 10 shows the annotated and the computed plots.

Argentina's GDP This dataset contains the GDP of Argentina in constant local currency, obtained from the World Bank. Figure 11 shows the annotated and the computed plots.

Iran's GDP This dataset contains the GDP of Iran in constant local currency, obtained from the World Bank. Figure 12 shows the annotated and the computed plots.

Japan's GDP This dataset contains the GDP of Japan in constant local currency, obtained from the World Bank. Figure 13 shows the annotated and the computed plots.

G.2. Precision and Recall

In this subsection, we report tables showing the precision and recall in the experiments performed, for a margin of 5 points. The QFF version of ADAGA for Run Log and Business Inventories uses the exact linear kernel, as described in the main paper. Precision scores for ZERO are exactly 1.0 because van den Burg & Williams (2020) consider the point with index 0 as a CP detected by all the algorithms. Since this is not informative w.r.t. the performance of the algorithm, this value is not shown in the tables. The values can be found in Tables 5, 6, 7, and 8.

G.3. Varying the Margin

In this subsection, we report tables of the F-1 scores computed with different margins. Again, the QFF version of ADAGA for Run Log and Business Inventories uses the linear kernel. This shows robustness of the evaluation scheme. The values can be found in Tables 9, 10, 11, and 12.

G.4. Default Hyperparameters

Table 14 reports, for each benchmark (except ADAGA), the best F-1 score obtained when tuning their parameters on a grid for each dataset separately. Here, we consider only the real-world datasets. This is highly *overestimating their performance*, and does not resemble a realistic scenario since we need the ground truth to evaluate the F-1 score. Conversely, ADAGA uses only one set of hyperparameters for all datasets. Nevertheless, ADAGA *still performs consistently well across all datasets*. Thus, we are confident that our hyperparameters can in principle be considered as good defaults. Furthermore, the grid search does not favor a particular set of hyperparameters for the competitors, leading to different values for each dataset, as shown in Table 13. Thus, this legitimates our comparison with the default hyperparameters of the benchmarks.

Table 13. Results of the grid search. The optimal hyperparameters are different for each dataset. This means that there is no unique candidate set of hyperparameters that can be used instead of the default ones, to improve the performance of the algorithms consistently.

ALGORITHM	RUN LOG	BUSINV	OZONE	GDP IRAN	GDP ARGENTINA	GDP JAPAN
BINSEG (mean)	SIC	MBIC	None	None	SIC	SIC
BINSEG (mean & var)	SIC	SIC	SIC	SIC	SIC	MBIC
PELT (mean)	MBIC	MBIC	SIC	MBIC	SIC	AIC
PELT (mean & var)	MBIC	SIC	MBIC	Hannan-Quinn	SIC	MBIC
BOCPD	$a = 0.001$	$a = 0.01$	$a = 0.1$	$a = 1.0$	$a = 0.001$	$a = 0.001$
	$b = 0.001$	$b = 0.0001$	$b = 0.1$	$b = 0.1$	$b = 0.0001$	$b = 1.0$
	$k = 0.01$	$k = 10.0$	$k = 0.01$	$k = 1.0$	$k = 10.0$	$k = 0.001$
RBOCPDMS	$a = 0.01$	$a = 1.0$	$a = 0.1$	$a = 0.01$	$a = 0.01$	$a = 0.01$
	$b = 0.0001$	$b = 0.0001$	$b = 0.0001$	$b = 0.0001$	$b = 0.0001$	$b = 0.01$
GPTS-CP (linear+const)	30	30	–	–	–	–
GPTS-CP (RQ+const)	–	–	45	15	30	30

Table 14. F-1 scores of ADAGA for CP detection and six benchmarks across six real-world datasets. A margin of 5 points was used. **All algorithms except ours have been overfit to the data**, by tuning their parameters to the best F-1 score individually for each dataset. **Despite this unrealistic and unfavorable comparison, ADAGA performs consistently well across all datasets**.

ALGORITHM	RUN LOG	BUSINV	OZONE	GDP IRAN	GDP ARGENTINA	GDP JAPAN	AVERAGE
ADAGA (exact, linear)	0.57	**0.77**	–	–	–	–	0.67
ADAGA (IPs, linear)	0.60	0.63	–	–	–	–	0.62
ADAGA (QFFs, RBF)	–	–	0.97	0.87	0.82	0.89	**0.89**
ADAGA (IPs, RBF)	–	–	0.78	0.80	0.89	0.62	0.77
ADAGA (IPs, Matern52)	–	–	0.97	0.80	0.82	0.89	0.87
ADAGA (IPs, RQ)	–	–	0.97	0.80	0.82	0.62	0.8
BINSEG (mean)	0.43	0.37	0.67	0.62	**0.95**	0.62	0.61
BINSEG (mean & var)	0.35	0.24	0.67	0.41	0.8	0.57	0.51
PELT (mean)	0.31	0.37	**1.0**	0.49	**0.95**	0.8	0.65
PELT (mean & var)	0.45	0.22	0.60	0.56	0.67	0.50	0.50
BOCPD	0.67	0.39	0.86	0.86	**0.95**	**1.0**	0.79
RBOCPDMS	0.56	0.56	0.78	0.49	0.58	0.8	0.63
GPTS-CP (linear+const)	**0.84**	0.62	–	–	–	–	0.78
GPTS-CP (RQ+const)	–	–	0.97	**0.97**	**0.95**	0.67	0.87

Table 15. Average F-1 scores scores (with standard deviation) of CUSUM for CP detection, across three synthetic datasets (10 noisy realizations per dataset). A margin of 5 points was used.

ALGORITHM	MEAN SHIFT DATA	VARIANCE SHIFT DATA	PERIODICITY SHIFT DATA	AVERAGE
CUSUM	0.4 ± 0	0.5 ± 0	0.5 ± 0	0.47 ± 0.05

Table 16. Average F-1 scores of CUSUM for CP detection, across six real-world datasets. A margin of 5 points was used.

ALGORITHM	RUN LOG	BUSINV	OZONE	GDP IRAN	GDP ARGENTINA	GDP JAPAN	AVERAGE
CUSUM	0.45	0.59	0.97	0.65	0.82	**0.89**	0.73

Grid search details:

- for BINSEG and PELT (mean, mean & var): the penalty was chosen from ["SIC", "BIC", "MBIC", "None", "AIC", "Hannan-Quinn"];
- for BOCPD: $a \in [0.001, 0.01, 0.1, 1.0, 10.0]$, $b \in [0.0001, 0.001, 0.01, 0.1, 1.0]$, $k \in [0.001, 0.01, 0.1, 1.0, 10.0]$;
- for RBOCPDMS: $a \in [0.01, 0.1, 1.0]$, $b \in [0.0001, 0.001, 0.01]$;
- for GPTS-CP: the GP hyperparameters are inferred from the first $[15, 30, 45]$ points.

H. Comparison with CUSUM Control Chart

Control charts constitute an interesting benchmark for our algorithm. We have processed our series with the Cumulative sum control chart (Page, 1954), using a public Python implementation[4], and we report the results in Tables 15 and 16 (with a margin of 5 points). In particular, for Run Log and Business Inventories datasets, we differentiate the time series, since we are interested in changes in trend of the series.

As the results show, CUSUM-based control chart is outperformed by our method, which can detect a wider class of CPs.

[4]The code is available at `https://github.com/BorgwardtLab/PyChange`.