Going from Offline to Online Change-Point Detection

Christiana Zorzi

A project submitted for the fulfillment of
the requirements of DSC 573

Under the supervision of Dr. Andreas Anastasiou

Department of Mathematics and Statistics
University of Cyprus
Fall Semester 2025

1 Introduction

Change-point detection refers to the process of identifying changes in the statistical properties of data over time. It has been an active area of research since the 1950s and has attracted significant interest due to its broad applicability across numerous fields. Its earliest applications were in quality control, and it was later used in areas such as environmental monitoring, biomedical signal processing, fault detection, and network security. More recently, its use has expanded to structural health monitoring and transportation systems, showing that change-point detection remains a flexible and widely applicable tool.

Change-point detection methods are generally divided into two main categories: offline (or a posteriori) methods, which analyze the entire dataset after all observations have been collected, and online (or sequential) methods, which detect changes as data are observed in real time. In the sequential setting, the process is monitored continuously, and action is taken only when a change occurs. The main challenge is to detect the change as quickly as possible while keeping false alarms to a pre-specified and feasible minimum. The first formal approach to this problem was introduced by Page (1954) through the 'Continuous inspection scheme', later known as the CUSUM method, which monitors data sequentially to detect changes as soon as they occur. Since then, various sequential algorithms have been developed to improve detection accuracy and flexibility across both Bayesian and frequentist frameworks. In the frequentist framework, the change-point is treated as an unknown, not necessarily random, number. Closely related to CUSUM is the Shiryaev-Roberts (S-R) method (Roberts, 1966), which also updates a running statistic to detect changes. The theoretical behaviour of such procedures was further clarified by Lorden (1971) and Pollak (1985), who proposed useful measures of detection speed and false alarms, showing why CUSUM and S-R methods are effective. Bayesian methods offer an alternative perspective by treating the change-point as a random variable with a prior distribution, allowing decisions to be based on the posterior probability that a change has occurred (see, for example Shiryaev (1963)); however, these are not the focus of the present work. One more recent method is the Change Point Model (CPM) (Ross, 2015), which provides both parametric and non-parametric methods for detecting changes in real time. In the sequential setting, a test statistic is updated with each new observation, and a change is flagged when it exceeds a predefined threshold. These thresholds are selected to control the false detection rate, typically measured by the Average Run Length (ARL), which represents the expected number of observations between false alarms when no real change has occurred.

In the offline setting, a variety of algorithms have been developed to detect multiple change-points once the entire dataset has been observed. The goal is to estimate the unknown number of change-points and their locations. Offline change-point detection techniques can be split into two main categories: those that detect all the change-points at once and those where the detection is carried out with one change-point at a time. The first category includes optimization-based methods, where the estimated signal is obtained by minimizing a least-squares or a log-likelihood cost function, penalized by a complexity term to prevent overfitting. An example from this category is the Segment Neighborhood method, by Auger and Lawrence (1989), which finds the globally opti-

mal segmentation through dynamic programming. Another example is the Pruned Exact Linear Time (PELT) method, by Killick et al. (2012), which improves efficiency by introducing a pruning rule. In the second category, where the change-points are detected one at a time, one popular method is Binary Segmentation (BinSeg) (Vostrikova, 1981), a top-down, hierarchical method which searches for a single, most prominent, change-point using an appropriate test statistic, splits the data at the detected point, and repeats the process on the resulting subintervals. Although computationally efficient, BinSeg can miss nearby change-points. To address such issues, Wild Binary Segmentation (WBS) (Fryzlewicz, 2014) extends the approach by randomly taking subintervals of the data, improving detection accuracy in cases with frequent or closely spaced changes. Further developments include WBS2 (Fryzlewicz, 2020), which refreshes its random intervals after each detected change-point, and the Narrowest-Over-Threshold (NOT) algorithm, by Baranowski et al. (2019), which selects the smallest interval whose contrast statistic exceeds a chosen threshold, ensuring that the interval contains a single change-point with high probability. Isolate-Detect (Anastasiou and Fryzlewicz, 2022) offers another approach for handling closely spaced change-points by isolating each candidate change-point before detection.

Among the existing offline methods, Isolate-Detect (ID), proposed by Anastasiou and Fryzlewicz (2022), stands out for its efficiency and strong detection performance. Its pseudo-sequential interval expansion character makes it an ideal candidate for extension to online change-point detection, forming the foundation of this research project. The Isolate-Detect algorithm detects changes by scanning both right- and left-expanding intervals that start from the beginning and the end of the data, respectively. Within each interval, it calculates a contrast statistic to identify potential changes. A change-point is detected when this statistic exceeds a predefined threshold. After each detection, the algorithm restarts on the remaining data, ensuring that the isolation of each change point is guaranteed. In cases where the threshold is misspecified, which may lead to a misestimation of the number of change points, Isolate-Detect addresses this by initially allowing for a potential overestimation of the number of detected changes. It then constructs a solution path, where the estimated change-points are ordered from the most to the least important, and selects the final model based on the optimization of a model selection criterion. However, in an online setting, data are constantly updated, and such solution paths cannot be constructed. Therefore, we focus on an extension of ID in the online setting through the threshold-based approach.

In this project, we aim to combine the two main categories of change-point detection, offline and online, by taking advantage of the sequential nature of the Isolate-Detect algorithm. Our goal is to modify and adapt it to operate effectively in an online environment.

As explained in Tartakovsky et al. (2014), online change-point detection problems can be categorized into two main scenarios: single-run detection procedures and multi-cyclic procedures. In the first scenario, the detection procedure is applied only once, and the outcome is either a false alarm or a correct detection that may occur with some delay. What happens after the stopping time is of no further interest. In the second scenario, the detection procedure is applied repeatedly, resetting after each detection from scratch, to allow continuous monitoring for future changes. In this project, we focus on the former case.

We will focus on the model

$$X_t = f_t + \sigma\epsilon_t, \quad t = 1, 2, \ldots \tag{1}$$

where X_t represents the observed value at time t, f_t is a one-dimensional, deterministic signal and ϵ_t are independent and identically distributed random variables. The signal f_t is assumed to be either piecewise-constant or continuous, piecewise-linear. The proposed algorithm, called Pseudo-Online Isolate-Detect (POID), operates in two main stages: expand and slide. It starts with a burn-in period during which the algorithm applies the existing Isolate-Detect method to learn the data pattern and establish a baseline. The proposed procedure then begins, after this burn-in phase, to monitor for new changes sequentially. Starting from the expand stage, the monitoring window gradually increases as new observations arrive. Once the pre-defined window length is reached, the algorithm transitions to the slide stage, during which a fixed-length window moves forward sequentially, as data arrive. If no change-point is flagged, the slide stage begins from the point immediately after the end of the burn-in period. Otherwise, it starts from the point immediately after the detected change-point. At each step, a modified version of the Isolate-Detect algorithm is applied to right-expanding and sliding windows, enabling online detection. The parameters that correspond to the gradual increase of the monitoring window, the burn-in period and the pre-defined window length are formally introduced in Section 2.2, and their selection is discussed in Section 3, where a corresponding simulation study is presented.

The report is organised as follows. In Section 2, we provide a more detailed description of the Isolate-Detect framework, and the methodology behind POID. Section 3 presents the simulation study conducted to select the hyperparameters used in the POID algorithm. Section 4 applies the method to real financial data, including a live-data example. The report concludes in Section 5 with a summary of the key findings and suggestions for future research.

2 Methodology

2.1 The Offline Isolate-Detect framework

The proposed method, POID, builds upon the Isolate-Detect algorithm of Anastasiou and Fryzlewicz (2022), originally designed for efficient multiple change-point detection in an offline context. The core idea of ID is to recursively isolate and check for potential change-points using local contrast statistics. To set the foundation for our online extension, we begin by discussing the offline framework in which the original ID algorithm operates.

ID considers a sequence of observations $\{X_t^{\mathrm{ID}}\}_{t=1}^{T_{\mathrm{ID}}}$ modeled as

$$X_t^{\mathrm{ID}} = f_t + \sigma\varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, 1),$$

where f_t is either a piecewise-constant signal or a continuous, piecewise-linear signal, with N unknown number of change-points $r_j, j = 1 \ldots N$, and T_{ID} the total length of the data sequence in which the ID algorithm is applied. The objective of the algorithm is

to estimate the location and the number, N, of these change-points. In the piecewise-constant case, a change-point corresponds to a shift in the mean of the signal. More formally:

$$f_t = \mu_j, \quad t = r_{j-1} + 1, \ldots, r_j, \quad \text{and} \quad f_{r_j} \neq f_{r_j+1}. \tag{2}$$

In the continuous, piecewise-linear case, a change corresponds to a shift in the slope, while continuity is maintained. Formally:

$$f_t = \mu_{j,1} + \mu_{j,2} t, \quad t = r_{j-1} + 1, \ldots, r_j, \tag{3}$$

with the additional continuity constraint $\mu_{k,1} + \mu_{k,2} r_k = \mu_{k+1,1} + \mu_{k+1,2} r_k$, which ensures that no jump occurs at the change-points. The change-points, r_k, satisfy $f_{r_k-1} + f_{r_k+1} \neq 2 f_{r_k}$.

The standard deviation, σ, of ε_t, if it is unknown, is estimated via the Median Absolute Deviation (MAD) method proposed by Hampel (1974), with the proposed estimator to be $\hat{\sigma} := \tilde{C} \times \text{median}(\,|\mathbf{x} - \text{median}(\mathbf{x})|\,)$, for $\mathbf{x} = (x_1, x_2, \ldots, x_{T_{\text{ID}}})$. Rousseeuw and Croux (1993) showed that for $\tilde{C} = 1.4826$, the proposed $\hat{\sigma}$ is a consistent estimator of the population standard deviation σ in the case of Gaussian data. This estimator is very robust, as evidenced by its 50% breakdown point. In practice, following Baranowski et al. (2019), the MAD estimator is applied to the first-order differenced data for piecewise-constant signals and to the second-order differenced data for continuous, piecewise-linear signals to limit the effect of potential change-points.

In the cases where f_t is a piecewise-constant signal, the chosen contrast function is the absolute value of the CUSUM statistic. That is $C^b_{s,e}(\mathbf{X}^{\text{ID}}) = \left|\tilde{X}^b_{s,e}\right|$, where

$$\tilde{X}^b_{s,e} = \sqrt{\frac{e-b}{n(b-s+1)}} \sum_{t=s}^{b} X_t^{\text{ID}} - \sqrt{\frac{b-s+1}{n(e-b)}} \sum_{t=b+1}^{e} X_t^{\text{ID}},$$

with $1 \leq s \leq b < e \leq T_{\text{ID}}$ and $n = e - s + 1$. In practice, the threshold is chosen to be

$$\zeta_{T_{\text{ID}}} = \hat{\sigma} \times C \times \sqrt{2 \log T_{\text{ID}}},$$

where $C = 1.15$. In this case, the standard deviation is estimated by

$$\hat{\sigma} := 1.4826 \times \text{median}(|\mathbf{Y} - \text{median}(\mathbf{Y})|)\,/\sqrt{2},$$

where $\mathbf{Y} = (X_2^{\text{ID}} - X_1^{\text{ID}}, \ldots, X_{T_{\text{ID}}}^{\text{ID}} - X_{T_{\text{ID}}-1}^{\text{ID}})$.

In the cases where f_t is a continuous, piecewise-linear signal, the contrast function is $C^b_{s,e}(\mathbf{X}^{\text{ID}}) = \left|\langle \mathbf{X}^{\text{ID}}, \boldsymbol{\phi}^b_{s,e} \rangle\right|$, where $\boldsymbol{\phi}^b_{s,e}$ is a contrast vector designed so that the contrast function is maximized at the same location as the generalized log-likelihood ratio statistic, $R^b_{s,e}(\mathbf{X}^{\text{ID}})$, under Gaussianity. Baranowski et al. (2019) show that $\boldsymbol{\phi}^b_{s,e} = (\phi^b_{s,e}(1), \ldots, \phi^b_{s,e}(T_{\text{ID}}))$, where

$$\phi^b_{s,e}(t) = \begin{cases} \alpha^b_{s,e}\beta^b_{s,e}\big[(e+2b-3s+2)t - (be+bs-2s^2+2s)\big], & t = s, \ldots, b, \\ -\frac{\alpha^b_{s,e}}{\beta^b_{s,e}}\big[(3e-2b-s+2)t - (2e^2+2e-be-bs)\big], & t = b+1, \ldots, e, \\ 0, & \text{otherwise}, \end{cases}$$

where $\alpha_{s,e}^b = (6/[n(n^2-1)(1+(e-b+1)(b-s+1)+(e-b)(b-s))])^{1/2}$, $\beta_{s,e}^b = ([(e-b+1)(e-b)]/[(b-s+1)(b-s)])^{1/2}$ and $n = e-s+1$. Similarly to the previous case, where f_t was a piecewise-constant signal, in practice the threshold is chosen to be

$$\zeta_{T_{\text{ID}}} = \hat{\sigma} \times C \times \sqrt{2 \log T_{\text{ID}}},$$

where $C = 1.4$ (Anastasiou and Fryzlewicz, 2022). In this case, the standard deviation is estimated by

$$\hat{\sigma} := 1.4826 \times \text{median}(|\mathbf{Y} - \text{median}(\mathbf{Y})|) / \sqrt{6},$$

where $\mathbf{Y} = (X_1^{\text{ID}} - 2X_2^{\text{ID}} + X_3^{\text{ID}}, \ldots, X_{T_{\text{ID}}-2}^{\text{ID}} - 2X_{T_{\text{ID}}-1}^{\text{ID}} + X_{T_{\text{ID}}}^{\text{ID}})$.

However, the standard formulation of ID assumes full access to the entire signal, making it unsuitable for sequential or streaming data, where observations arrive over time. In this project, we adapt the ID framework for real-time monitoring by developing an expand-and-slide strategy that enables online change-point detection while retaining ID's contrast-based structure and theoretical strengths.

2.2 Sequential Detection Procedure

We work under the model in (1). POID extends the Isolate-Detect framework to an online, single-run setting, where observations arrive sequentially, and the aim is to detect a change as soon as possible, after it occurs.

To enable sequential detection, several hyperparameters are introduced in order to control how the algorithm monitors and updates the data stream. The window length $W \in \mathbb{N}$ determines the maximum number of observations used in each monitoring window. The step ahead, denoted by $\lambda \in \mathbb{N}$, specifies the number of new observations added sequentially before the next evaluation of the monitoring window. The threshold signal length, which is the length of the signal used in the detection threshold, $\zeta_T \in \mathbb{R}^+$, is denoted by $T \in \mathbb{N}$. Together, these parameters balance the detection speed and the accuracy, and their optimal values are chosen from a simulation study discussed in Section 3.

The algorithm begins by applying the existing Isolate-Detect algorithm at a predefined burn-in period of T_{bi} data points. The purpose of this phase is to allow the algorithm to learn the initial behaviour of the signal before the monitoring begins. In this project, we set $T_{bi} = 1000$, which provides a sufficiently long initialization period. Other values could also be considered, as this choice does not affect the overall logic of the algorithm. If change-points are detected within the burn-in period, the procedure starts from the data point immediately following the last detected change-point. If no change-point is detected, it starts from the point immediately after the end of the burn-in period. We denote this starting point by $s \in \mathbb{N}$.

After this initialization, the algorithm enters the expanding stage. At this stage, the modified Isolate-Detect algorithm is applied to the expanding intervals that grow as new data arrive. This modification of ID is minimal but necessary to accommodate the online setting. In the original ID framework, both left- and right-expanding intervals are used to isolate and detect potential change-points within the full, fixed-length dataset. However, in a sequential context, data become available only in one direction, since new observations are appended to the right of the existing stream. To account for this, the

original ID algorithm was adapted to operate with right-only expanding intervals, enabling real-time monitoring. At each step, the monitoring window expands by λ new data points that arrive sequentially, before the modified ID algorithm is reapplied. In this project, we consider $\lambda \in \{1, 3, 10\}$. Therefore, the modified ID algorithm is applied to the current interval $[s, e]$, where s is the starting data point and is fixed, and e increases as new data arrive, every time by λ. This process continues until a change point is detected, i.e. $C^b_{s,e}(\mathbf{X}) > \zeta_T$, for some $b \in [s, e)$ or until the end of the window-length, W, is reached. If a change-point is detected, the procedure terminates. The values of W considered are 500 and 1000 data points. The threshold signal length T, is chosen to be constant across all monitoring intervals rather than increasing with $e - s$. This choice ensures that the decision rule remains consistent throughout the expanding stage. With $C = 1.15$ for the piecewise-constant signals, and $C = 1.4$ for the continuous, piecewise-linear signals, as explained in Section 2.1, the threshold $\zeta_T = \hat{\sigma} \times C \times \sqrt{2 \log T}$ determines how easily a change-point is detected. A smaller value of T decreases the threshold ζ_T, making it easier for the contrast $C^b_{s,e}(\mathbf{X})$ to exceed this threshold. As a result, the detection of a change-point becomes easier, which increases the risk of raising a false alarm. In contrast, a larger value of T raises the threshold, making the procedure more conservative. This reduces the chance of having a false alarm, but increases the chance of missing a true change-point. During the expanding phase, the monitoring interval $[s, e]$ naturally grows over time and if T was allowed to increase with it, the threshold would also increase, and therefore changing the behaviour of the procedure. Keeping T constant avoids this issue and ensures that detections are judged under the same conditions. At each step, the noise standard deviation, $\hat{\sigma}$, used for calculating the threshold, is re-estimated using the most recent portion of the data. Specifically, the estimate is computed from the last 100 observations. This local re-estimation allows the algorithm to adjust to potential changes in noise level as new data become available.

If the maximum window length W is reached without detecting a change, the algorithm transitions to the sliding stage. In this stage, the monitoring window is fixed to the maximum length W, and the algorithm continues to scan the data by moving this window forward as new observations become available. At each step, the window slides by λ data points, the same step-ahead parameter used during the expanding phase. The algorithm applies the modified ID to consecutive intervals $[s', e']$, where both the starting and the ending points are updated as $s' = s + \lambda$ and $e' = e + \lambda$. This ensures that the window moves along the data stream, analyzing the same number of observations at each step. As in the expanding stage, the noise standard deviation, $\hat{\sigma}$, is re-estimated at each step using the most recent 100 data points. A change-point is declared once $C^b_{s',e'}(\mathbf{X}) > \zeta_T$, for some $b \in [s', e')$, and then, the procedure terminates. The optimal values of the window length W, step ahead parameter λ and threshold signal length T are determined through the simulation study presented in Section 3.

3 Simulations

3.1 Simulation study

In this section, we present a simulation study that was conducted for the parameter calibration of the proposed sequential change-point detection algorithm, POID. The algorithm's performance depends on parameters such as the signal length, T, used for the computation of the threshold, the window size W, and the step-ahead, λ, size, as introduced in Section 2.2. We therefore conduct a series of simulations to explore how these choices influence detection accuracy, delay, and false alarm rate. The aim is to identify a combination of parameters that provides a good balance between fast and accurate detection. Both for the change in the mean and the change in the slope, we used signals with a total length (T_{tot}) of 3000 points. However, this simulation study is independent of T_{tot}, since we imitate an online setting.

Two sets of signals were generated to assess the algorithm's ability to detect changes in the mean. The first set consisted of seven signals of total length 3000, each containing two change-points located at positions 300 and 1500. The second set, consisted again of seven signals with the same total length but with change-points at locations 900 and 1050. More specifically, for the first set of signals, we have:

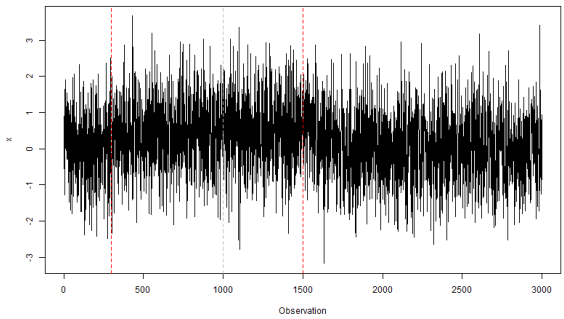
$$f(t) = \begin{cases} 0, & 1 \leq t \leq 300, \\ \delta, & 301 \leq t \leq 1500, \\ 0, & 1501 \leq t \leq 3000 \end{cases} \tag{4}$$

and for the second set:

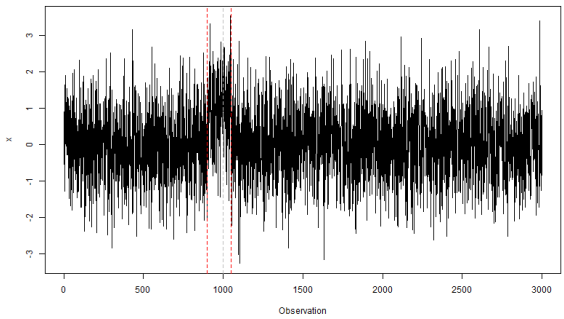
$$f(t) = \begin{cases} 0, & 1 \leq t \leq 900, \\ \delta, & 901 \leq t \leq 1050, \\ 0, & 1051 \leq t \leq 3000 \end{cases} \tag{5}$$

where $\delta \in \{0.5, 1, 2, 3, 5, 8, 10\}$ represents the magnitude of the mean shift. Examples can be found in Figure 1. Tables 1-4 show the results of the simulations of the first case shown in (4), while Tables 5-8 show the results of the simulations of the second case shown in (5).

Similarly to the case of change in the mean, we generate two sets of signals to assess the algorithm's ability to detect changes in the slope. The first set consisted of seven signals of total length 3000, each containing two change-points located at positions 300 and 1500. Therefore, based on the model (3), $r_1 = 300$ and $r_2 = 1500$. The second set consisted of seven signals with the same total length but with change-points at locations 900 and 1050. Therefore, $r_1 = 900$ and $r_2 = 1050$, based on the model (3). The above predefined change-point locations split the signals into three linear segments. The signals are constructed such that the first and third segments share the same slope a, while the middle segment has slope $b = 1$, with continuity enforced at each change-point. Examples can be found in Figure 2. Specifically, each signal in the first case is defined as:



(a) Change-points at 300 and 1500



(b) Change-points at 900 and 1050

Figure 1: Change in mean examples with $\delta = 1$ and $T_{bi} = 1000$

$$f(t) = \begin{cases} a \cdot t, & 1 \leq t \leq 300, \\ a \cdot 300 + b(t - 300), & 301 \leq t \leq 1500, \\ a \cdot 300 + b(1500 - 300) + a(t - 1500), & 1501 \leq t \leq 3000, \end{cases} \tag{6}$$

where

$$a \in \{3,\ 2.7,\ 2.5,\ 2,\ 1.7,\ 1.5,\ 1.2\}, \quad b = 1,$$

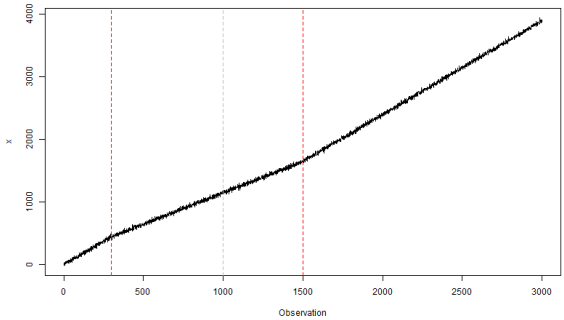
represent the slopes before and after the change-points, respectively.

Similarly, for the second case we have:

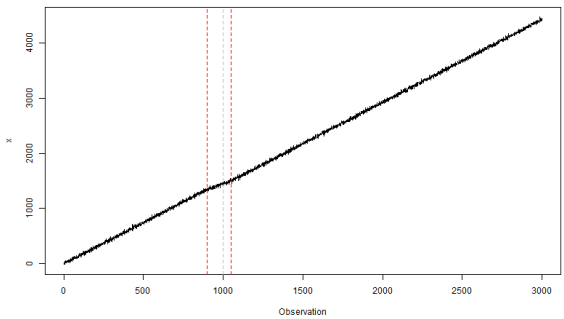
$$f(t) = \begin{cases} a \cdot t, & 1 \leq t \leq 900, \\ a \cdot 900 + b(t - 900), & 901 \leq t \leq 1050, \\ a \cdot 900 + b(1050 - 900) + a(t - 1050), & 1051 \leq t \leq 3000, \end{cases} \tag{7}$$

where

$$a \in \{3,\ 2.7,\ 2.5,\ 2,\ 1.7,\ 1.5,\ 1.2\}, \quad b = 1.$$



(a) Change-points at 300 and 1500



(b) Change-points at 900 and 1050

Figure 2: Change in slope examples with $a = 1.5$, $b = 1$ and $T_{bi} = 1000$

To assess the performance of POID and allow comparisons between different combinations of parameters, we used several performance metrics that capture the detection accuracy. More specifically, for each simulated signal, the following metrics were computed over 100 replications:

1. **Number of correct detections**: the number of detected change-points that fall within ± 20 observations from the true change-point location.

2. **Number of missed detections**: cases where no change-point was detected within the specified correct detection range.

3. **Number of wrong detections**: cases where the algorithm signals a change at a current time earlier than the true change-point.

4. **Mean and median delay**: the average and median for the distance between the current time and the detected change-point, where the current time is after the time of the detected change-point. We consider these detections, valid detections.

5. **Median location difference**: the median number of observations between the estimated and the true change-point location, in absolute value, in valid detections, as explained in the mean and median delay above.

6. **Median of the normalized location error**: computed as

$$\text{Median}\left(\frac{|\hat{\tau}-\tau|}{T_{tot}}\right),$$

   where $\hat{\tau}$ is the estimated change-point, τ is the true change-point, and T_{tot} is the total signal length. This metric measures the detection precision relative to the total length of the signal.

The combinations of parameters used in both change in the mean and change in the slope are:

- **Window length** (W): the size of the moving window. Two values were examined, 500 and 1000.

- **Step ahead size** (λ): the number of new observations added at each step. The values of λ we tested are 1, 3, and 10.

- **Threshold signal length** (T): the signal length used in the computation of the detection threshold. The values of T that we tested are 500 and 1000.

Each combination of these parameters was applied to all simulated signals, and the performance metrics were averaged over 100 independent replications. In all simulation scenarios, both for changes in the mean and changes in the slope, the first change-point was placed within the burn-in period. This change-point is detected by the ID existing algorithm, before our sequential algorithm begins. In case ID detects more than one change-point in the burn-in period, then the sequential detection procedure starts from

the data point immediately after the last detected change-point. In case ID finds no change-points, it starts from the point immediately after the end of the burn-in period. Including additional or no change-points within this burn-in period would not provide further insight, as such cases do not affect the logic of the procedure. The aim is to detect the first, and only, change-point after the burn-in period, as the data arrive sequentially. The results are reported in the tables 1-8, for change in the mean, and in the tables 9-16, for change in the slope, allowing for a direct comparison across different combinations of the parameters.

3.2 Results for change in the mean

Starting from the case where the change occurs in the mean of the signal, we evaluate the performance of POID under different magnitudes of mean shifts. The simulated signals are constructed according to model (1), where the function f_t has sudden level shifts at predefined locations while the noise variance is kept constant at $\sigma^2 = 1$. The aim of this experiment is to assess how effectively the algorithm can detect such changes, as well as to investigate the influence of the hyperparameters window length W, step ahead size λ and threshold signal length T.

Tables 1, 2, 3 and 4 show the results of the simulations of the first case shown at (4).

Table 1: Simulation results for change in mean with $T = 1000$, $W = 1000$ and $\lambda = 10$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.5	62	0	9	63	72.68	10	0.003
1	87	0	9	22	22.97	2	0.001
2	91	0	9	10	10.68	0	0
3	91	0	9	10	9.96	0	0
5	91	0	9	10	9.21	0	0
8	91	0	9	10	9.20	0	0
10	91	0	9	10	9.20	0	0

Table 2: Simulation results for change in mean with $T = 1000$, $W = 1000$ and $\lambda = 3$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.5	61	0	10	60	69.14	10	0.003
1	87	0	9	19	20.19	1	0.0003
2	92	0	8	7	7.01	0	0
3	92	0	8	4	4.52	0	0
5	92	0	8	4	3.83	0	0
8	92	0	8	4	3.88	0	0
10	92	0	8	4	3.88	0	0

Table 3: Simulation results for change in mean with $T = 500$, $W = 1000$ and $\lambda = 1$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.5	50	0	23	48	56.74	8	0.003
1	74	0	23	15	16	1	0.0003
2	77	0	23	5	5.34	0	0
3	77	0	23	3	3.32	0	0
5	77	0	23	2	2.13	0	0
8	77	0	23	2	2	0	0
10	77	0	23	2	2	0	0

Table 4: Simulation results for change in mean with $T = 500$, $W = 500$ and $\lambda = 10$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.5	48	0	30	59	74.17	9.5	0.003
1	70	0	30	19	22.03	1	0.0003
2	69	0	30	9	10.03	0	0
3	69	0	30	9	9.03	0	0
5	70	0	30	9	8.89	0	0
8	70	0	30	9	8.89	0	0
10	70	0	30	9	8.89	0	0

Tables 5, 6, 7 and 8 show the results of the simulations of the second case shown at (5).

Table 5: Simulation results for change in mean with $T = 1000$, $W = 1000$ and $\lambda = 10$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.5	63	5	0	82	148	10	0.003
1	95	0	1	22	24.73	2	0.001
2	99	0	1	12	11.8	0	0
3	99	0	1	12	10.89	0	0
5	99	0	1	2	4.53	0	0
8	99	0	1	2	2	0	0
10	99	0	1	2	2	0	0

Table 6: Simulation results for change in mean with $T = 1000$, $W = 1000$ and $\lambda = 3$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.5	63	5	0	78	143.68	10	0.003
1	96	0	1	18	21.3	2	0.001
2	99	0	1	6	6.88	0	0
3	99	0	1	3	4.39	0	0
5	99	0	1	3	3	0	0
8	99	0	1	3	3	0	0
10	99	0	1	3	3	0	0

Table 7: Simulation results for change in mean with $T = 500$, $W = 1000$ and $\lambda = 1$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.5	60	3	2	61	106.82	11	0.004
1	94	0	2	16	18.01	1.5	0.001
2	98	0	2	5	5.55	0	0
3	98	0	2	3	3.33	0	0
5	98	0	2	2	2.19	0	0
8	98	0	2	2	2	0	0
10	98	0	2	2	2	0	0

Table 8: Simulation results for change in mean with $T = 500$, $W = 500$ and $\lambda = 10$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.5	59	1	2	62	140.82	13	0.004
1	94	0	2	22	22.31	1.5	0.001
2	98	0	2	12	11.69	0	0
3	98	0	2	12	10.37	0	0
5	98	0	2	2	3.94	0	0
8	98	0	2	2	2	0	0
10	98	0	2	2	2	0	0

The results presented in the above tables show that POID performs well in identifying the changes in the mean across a wide range of signal magnitudes. As expected, in all cases, smaller change magnitudes have less correct detections and higher delays. Regarding the changes in the parameters, we can clearly see that higher values of the window length, W, and the threshold signal length, T, lead to more accurate detections. The choice of the step ahead, λ, affects the detection delay, with lower values producing detections with a lower delay at the cost of higher computational effort. For small change magnitudes, for example $\delta = 0.5$ or 1, we see that the algorithm struggles to detect the change, resulting in longer delays and occasional missed detections. We can also see that at the second set of the signals, results shown in Tables 5-8, where the true location of the change-point was near the end of the burn-in period, the detection accuracy was higher, even in smaller values of the length of the threshold T, compared to the first set of signals. This is due to the fact that the threshold function in lower values of T is less conservative, meaning that it detects changes easier, and therefore it increases the chance of getting a false-alarm. Since 1050 is much closer to 1000, which is the end of the burn-in period, than 1500, the algorithm in the first set of signals does more checks until it detects the change point, compared to the second case, which leads to a higher false positive rate, and explains the higher number of wrong detections. Generally, the algorithm seems to have a good overall performance and accurate detection of the change-points under different change magnitudes and parameter combinations.

From the above results, the parameter combination with window length $W = 1000$, step ahead $\lambda = 3$, and threshold signal length $T = 1000$ was chosen as the best-performing setup, as it provides an effective trade-off between accuracy, delay, and computational cost.

3.3 Results for change in the slope

Moving to the case where we have continuous, piecewise-linear signals, the aim is to evaluate the algorithm's ability to detect changes in the slope of the underlying trend. The simulated signals are again constructed according to model (1), where f_t is a continuous function with changes occurring in its first derivative, while the noise variance is kept constant at $\sigma^2 = 400$.

Tables 9, 10, 11 and 12 show the results of the simulations of the first case shown at (6).

Table 9: Simulation results for change in slope with $T = 1000$, $W = 1000$ and $\lambda = 10$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.2	59	0	0	105	104.67	15.5	0.005
0.5	86	0	0	58	56.97	8	0.003
0.7	91	0	0	45	45.75	6.5	0.002
1.0	95	0	0	37	36.32	5.5	0.002
1.5	96	0	0	28.5	28.34	4	0.001
1.7	97	0	0	27	26.54	4	0.001
2.0	98	0	0	24.5	24.36	3	0.001

Table 10: Simulation results for change in slope with $T = 1000$, $W = 1000$ and $\lambda = 3$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.2	60	0	0	102	100.38	16.5	0.006
0.5	82	0	0	53.5	53.47	8	0.003
0.7	90	0	0	42	42.24	6	0.002
1.0	95	0	0	33	33.02	5	0.002
1.5	96	0	0	25	24.74	4	0.001
1.7	98	0	0	23	22.96	4	0.001
2.0	98	0	0	21	20.52	3	0.001

Table 11: Simulation results for change in slope with $T = 500$, $W = 1000$ and $\lambda = 1$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.2	53	0	0	95	93.78	20	0.007
0.5	79	0	0	50	50.22	8	0.003
0.7	91	0	0	40	39.6	7	0.002
1.0	95	0	0	30	30.32	5	0.002
1.5	97	0	0	23	23.04	4	0.001
1.7	98	0	0	21	21.03	4	0.001
2.0	98	0	0	19	18.96	3	0.001

Table 12: Simulation results for change in slope with $T = 500$, $W = 500$ and $\lambda = 10$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.2	56	0	1	109	115.16	18	0.006
0.5	79	0	1	59	58.29	9	0.003
0.7	88	0	1	49	46.27	7	0.002
1.0	94	0	1	39	36.98	5	0.002
1.5	95	0	1	29	28.7	4	0.001
1.7	98	0	1	29	26.58	4	0.001
2.0	98	0	1	29	24.15	4	0.001

Tables 13, 14, 15 and 16 show the results of the simulations of the second case shown at (7).

Table 13: Simulation results for change in slope with $T = 1000$, $W = 1000$ and $\lambda = 10$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.2	39	40	0	218	275.23	17	0.006
0.5	83	0	0	78	76.3	10	0.003
0.7	88	0	0	53	57.2	8	0.003
1.0	95	0	0	43	43.2	6	0.002
1.5	99	0	0	33	32.3	5	0.002
1.7	99	0	0	33	29.8	4	0.001
2.0	99	0	0	23	26.8	3	0.001

Table 14: Simulation results for change in slope with $T = 1000$, $W = 1000$ and $\lambda = 3$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.2	39	40	0	215.5	271.53	17	0.006
0.5	83	0	0	74.5	73.42	10	0.003
0.7	88	0	0	53.5	53.89	8	0.003
1.0	95	0	0	40	40.09	6	0.002
1.5	99	0	0	28	28.87	5	0.002
1.7	99	0	0	28	26.8	4	0.001
2.0	99	0	0	22	23.59	3	0.001

Table 15: Simulation results for change in slope with $T = 500$, $W = 1000$ and $\lambda = 1$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.2	39	36	0	221.5	314.95	18	0.006
0.5	83	0	0	73.5	72.42	10	0.003
0.7	88	0	0	52.5	52.91	8	0.003
1.0	95	0	0	39	38.94	6	0.002
1.5	99	0	0	28	27.79	5	0.002
1.7	99	0	0	26	25.7	4	0.001
2.0	99	0	0	22	22.62	3	0.001

Table 16: Simulation results for change in slope with $T = 500$, $W = 500$ and $\lambda = 10$.

Change	Correct	No det.	Wrong	Median del.	Mean del.	Median diff.	Median norm. err.
0.2	34	45	0	213	247.15	17	0.006
0.5	83	0	0	78	76.3	10	0.003
0.7	88	0	0	53	57.2	8	0.003
1.0	95	0	0	43	43.2	6	0.002
1.5	99	0	0	33	32.3	5	0.002
1.7	99	0	0	33	29.8	4	0.001
2.0	99	0	0	23	26.8	3	0.001

From the above results, we again observe that POID performs very well in detecting changes in the slope across different signals. It is once again clear and expected that for higher change magnitudes, we have a higher detection accuracy with very small median delays and almost no location errors. For smaller slope changes, the number of correct detections decreases noticeably, and the detection delay increases significantly, reaching over 200 observations for the weakest changes in some occasions. This behaviour is expected, as small changes in slope are harder to detect when the signal-to-noise ratio is low. Comparing across different parameter combinations, the results show that using a smaller λ slightly improves detection speed, reducing both the median and mean delay, while larger steps lead to longer detection times. Larger values of the window length W also give slightly more accurate detections. Overall, the algorithm seems to perform reliably across all settings, balancing detection speed and accuracy.

From the above results, the parameter combination with window length $W = 1000$, step ahead $\lambda = 3$, and threshold signal length $T = 1000$ was chosen as the best-performing setup, as it provides an effective trade-off between accuracy, delay, and computational cost.

4 Real Data

4.1 Apple Stock prices

To evaluate the behaviour of the proposed Pseudo-Online Isolate-Detect algorithm under realistic conditions, we apply it to real-time financial data obtained from the Twelve Data API (Twelve Data, 2025). This source provides real-time intraday stock price data, and for this project we use the Apple Inc. stock price. In this setting, our aim is to detect the first change in slope within the evolving price process. By requesting one observation at a time, the algorithm receives each new price as it becomes available and updates its monitoring window accordingly. This allows POID to operate in a real online setting, using only information available up to a current time, and declare change-points as soon as they occur.

The hyperparameter combination used is the one chosen as the best-performing, from the simulation study conducted in Section 3.3. Specifically, the window length, the step ahead parameter, and the threshold signal length are set to $W = 1000$, $\lambda = 3$ and $T = 1000$, respectively. The burn-in period was set to 150 data points.

The live data were collected on 09/12/2025, with a new observation retrieved every

7.5 seconds. The resulting price sequence is displayed in Figure 3. In this figure, the black line represents the incoming time series of the Apple stock price (in dollars), as it becomes available in real time. The gray vertical live represents the end of the burn-in period. The green vertical line represents the location of the detected change-point, while the blue vertical line represents the time of detection. The results show that the algorithm identifies the first clear drop in the price after the burn-in period, even though the movement is small. The delay between the estimated change-point and the time of its detection is very short; more specifically, the estimated change-point occurred at 20:41:05 EET and it was detected at 20:41:12 EET. This suggests that the procedure is able to react quickly once the underlying trend begins to change, requiring only a small number of post-change observations to accumulate sufficient evidence. The behaviour of the series after detection confirms that indeed a change has occurred.

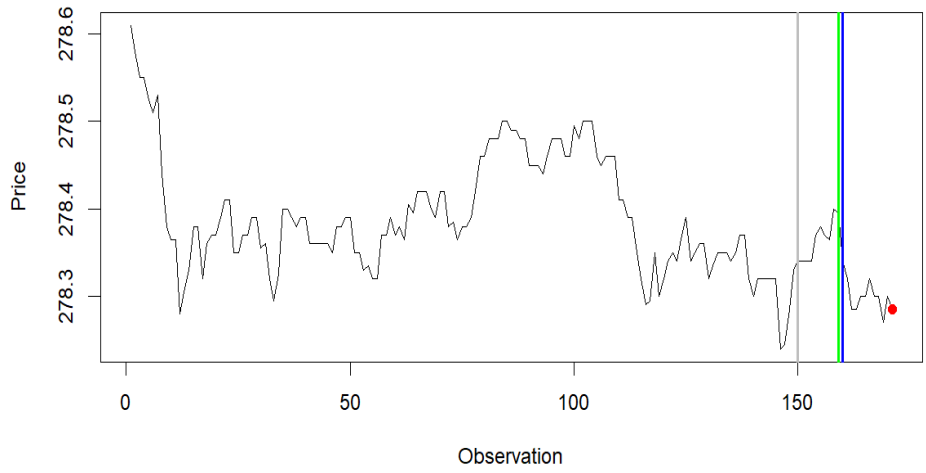


Figure 3: Live Apple stock prices on 09/12/2025, collected every 7.5 seconds

4.2 Euro to British Pound exchange rate

Here, we apply the POID algorithm to the Euro to British Pound exchange rate during the Brexit referendum in 2016, with the aim of detecting the first change in the mean level of the series. The historical daily closing exchange rate prices were obtained using the tidyquant package in R (see Dancho and Vaughan (2025) for more details), and the analysis begins in late November 2015 to ensure that the procedure receives a sufficient number of pre-referendum observations for the burn-in period, which is set to 150 data points. The data are supplied to the algorithm sequentially, so that the procedure operates in the same way as in an online setting. The hyperparameter combination used in the algorithm was $W = 1000$, $\lambda = 3$ and $T = 1000$, as chosen from the simulation study discussed in Section 3.2

Figure 4 displays the exchange rate series. Similarly to the previous example, the gray line indicates the end of the burn-in period, the green line marks the estimated mean shift, and the blue line marks the time at which the algorithm detects it. The

estimated change-point corresponds to 23 June 2016, the date of the Brexit referendum, and detection occurs a few days later, on 28 June 2016. This means that, if the algorithm had been running in real time, it would have detected the shift within approximately five trading days of its occurrence, using only the daily closing prices. The behaviour of the series after detection is consistent with a higher mean level, indicating that the algorithm successfully identifies the main structural change associated with the referendum.

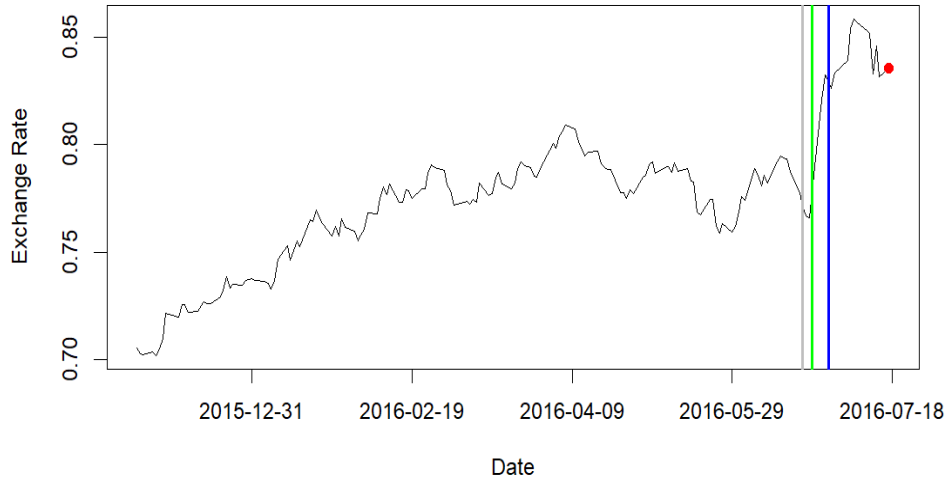


Figure 4: Daily Euro to British Pound exchange rate data for 2016, starting from 25/11/2015

5 Conclusions

In this project, we developed an online extension of the Isolate-Detect algorithm for real time change-point detection. The proposed Pseudo-Online Isolate-Detect algorithm takes advantage of the pseudo-sequential interval expansion character of the Isolate-Detect, and makes it work in a sequential setting through an expand-and-slide procedure, enabling the detection of a change as observations arrive in real time.

A simulation study was conducted for both changes in the mean and changes in the slope. The results show that POID detects moderate and large changes reliably, with high accuracy and small location error. For very small changes, as expected, detection becomes more difficult and delays increase. Both for the change in the mean and the change in the slope, the combination of window length, W, set to 1000, the step ahead parameter, λ, set to 3 and the threshold signal length, T, set to 1000, provided the best trade-off between accuracy, detection delay, and computational cost.

The algorithm was then applied to real financial data. Using live intraday Apple stock prices, POID successfully identified a small downward slope change with only a short detection delay, illustrating its ability to operate in a real sequential environment. A second application to the Euro to British Pound exchange rate, around the 2016 Brexit referendum, demonstrated that POID can also capture structural changes in historical

economic time series when these are processed one observation at a time. In both cases, the data after the detection supported the detected change.

While the results are encouraging, several limitations remain. Very small structural changes, particularly in noisy series, may require a larger number of observations before they can be reliably detected, which naturally increases the detection delay.

In future work, we could also explore multi-cyclic online monitoring, in which the algorithm repeatedly restarts after each detected change, allowing for ongoing monitoring with automatic reset. Another direction is the development of a real time dashboard that allows users to select the threshold signal length T according to their risk preferences, and proceeds to decision making. This would be particularly valuable in financial applications in which, for example, a trader could choose a smaller threshold signal length to detect changes earlier, accepting a higher probability of false-alarms, or a larger threshold for a more conservative strategy that reacts only to more substantial structural changes. The methodology could also be extended to multivariate time series, where identifying simultaneous or near-simultaneous structural changes across multiple variables is important. Beyond financial data, POID has the potential to be applied in any domain where real time detection is crucial and beneficial, such as network or health monitoring. Overall, POID may bridge the gap between offline methods and practical real-time monitoring tools.

References

Anastasiou, A. and Fryzlewicz, P. (2022). Detecting multiple generalized change-points by isolating single ones. *Metrika*, 85(2):141–174.

Auger, I. E. and Lawrence, C. E. (1989). Algorithms for the optimal identification of segment neighborhoods. *Bulletin of Mathematical Biology*, 51(1):39–54.

Baranowski, R., Chen, Y., and Fryzlewicz, P. (2019). Narrowest-over-threshold detection of multiple change points and change-point-like features. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 81(3):649–672.

Dancho, M. and Vaughan, D. (2025). *tidyquant: Tidy Quantitative Financial Analysis*. R package version 1.0.11.

Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243–2281.

Fryzlewicz, P. (2020). Detecting possibly frequent change-points: Wild binary segmentation 2 and steepest-drop model selection. *Journal of the Korean Statistical Society*, 49(4):1027–1070.

Hampel, F. R. (1974). The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

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

Lorden, G. (1971). Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42(6):1897–1908.

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

Pollak, M. (1985). Optimal detection of a change in distribution. *The Annals of Statistics*, 13(1):206–227.

Roberts, S. W. (1966). A comparison of some control chart procedures. *Technometrics*, 8(1):411–430.

Ross, G. J. (2015). Parametric and nonparametric sequential change detection in r: The cpm package. *Journal of Statistical Software*, 66(3):1–20.

Rousseeuw, P. J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283.

Shiryaev, A. N. (1963). On optimum methods in quickest detection problems. *Theory of Probability and its Applications*, 8(3):22–46.

Tartakovsky, A., Nikiforov, I., and Basseville, M. (2014). *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. Chapman & Hall/CRC, 1st edition.

Twelve Data (2025). Twelve data api documentation. https://twelvedata.com.

Vostrikova, L. Y. (1981). Detecting 'disorder' in multidimensional random processes. *Doklady Akademii Nauk*, 259(2):270–274.