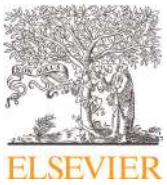
Future Generation Computer Systems 177 (2026) 108233



Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A kernel-based approach for accurate steady-state detection in performance time series

Martin Beseda *, Vittorio Cortellessa, Daniele Di Pompeo, Luca Traini, Michele Tucci

University of L'Aquila, Via Vetoio, 1, L'Aquila, 67100, Italy

ARTICLE INFO

Keywords:
Steady state
Convolution
Kernel
Time series
Warm up
Performance

ABSTRACT

This paper addresses the challenge of accurately detecting the transition from the warmup phase to the steady state in performance metric time series, which is a critical step for effective benchmarking. The goal is to introduce a method that avoids premature or delayed detection, which can lead to inaccurate or inefficient performance analysis. The proposed approach adapts techniques from the chemical reactors domain, detecting steady states online through the combination of kernel-based step detection and statistical methods. By using a window-based approach, it provides detailed information and improves the accuracy of identifying phase transitions, even in noisy or irregular time series. Results show that the new approach reduces total error by 14.5 % compared to the best selected state-of-the-art method. It offers more reliable detection of the steady-state onset, delivering greater precision for benchmarking tasks. For users, the new approach enhances the accuracy and stability of performance benchmarking, efficiently handling diverse time series data. Its robustness and adaptability make it a valuable tool for real-world performance evaluation, ensuring consistent and reproducible results.

1. Introduction

Time series of performance metrics collected on running systems are notoriously subjected to a two-phase behavior: (i) in the first phase, namely *warm-up*, the metric trend is irregular, mainly due to instability of resource assignment and/or runtime optimizations on the executed code; (ii) in a second phase, namely *steady state*, the collected values of metrics may achieve a certain degree of regularity over time. For obvious reasons, the warm-up section of the time series is excluded from the analysis of collected data.

A major (still unsolved) problem in performance benchmarking/testing is accurately identifying the phase change in a time series, i.e., the transition from warm-up to steady state, as the one illustrated on Fig. 1. The complexity of this task is due to several factors, like transient stable behaviors that can be erroneously interpreted as achieved steady states, or odd distributions of outliers across the whole time series. However, this task remains crucial for the effectiveness of performance benchmarking/testing. Indeed, a too-early identification of the phase change leads to the risk of accounting for unstable values in the analysis process, whereas a too-late identification leads to missing accurate values and/or spending useless additional testing time to achieve an adequate amount of stable measures for the analysis process. Despite its complexity, several approaches have been introduced in the last few years for addressing this problem, with different degrees of accuracy and effectiveness, ranging from statistical approaches to machine learning methods. Techniques like changepoint detection algorithms [1,2] or hybrid models combining statistical tests [3–5] have seen increasing use in benchmarking scenarios. However, accurately capturing the phase transition remains an open issue, particularly when faced with irregular time series or noise-heavy environments, as is common in system performance metrics.

In this paper, we consider a modification of approaches that are traditionally adopted for Steady-State Detection (SSD) in time series from the chemical reactor domain [6]. As such, the presented method Kernel-based Kelly's Steady State Detection (KB-KSSD) is able to monitor steadiness in an "online" manner and to provide some information on the "window" basis, i.e., the user can obtain more detailed information about the character of the analyzed time series.

Our primary objective with this work is not to propose a universally novel algorithm for generic time series analysis, but rather to enhance performance engineering practices by developing a method that works effectively and reliably in the domain of benchmark execution

* Corresponding author.
E-mail address: martin.beseda@univaq.it (M. Beseda).

https://doi.org/10.1016/j.future.2025.108233
Received 10 April 2025; Received in revised form 7 August 2025; Accepted 1 November 2025
Available online 5 November 2025

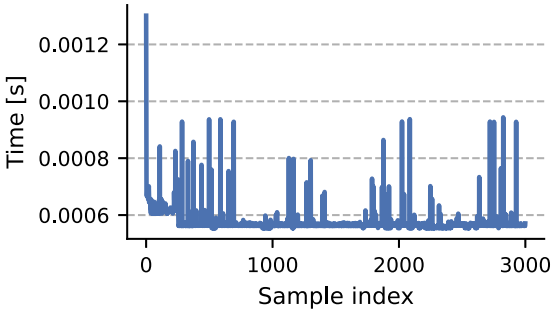


Fig. 1. Example of a performance timeseries with the clearly visible warm-up phase.

times. Indeed, we contribute several novel ideas and combinations that are tailored to how practitioners interpret performance data. More specifically:

- Our preprocessing approach tries to mimic how performance analysts typically assess noisy performance charts. We replace percentile-based outliers with local medians, followed by median-based filtering, to ensure that transient anomalies do not lead to incorrect conclusions about steadiness. This is essential in performance engineering, where single outliers can skew interpretations.
- Rather than relying on purely statistical changepoint detection, our approach uses convolution with asymmetric kernels to directly look for step-down patterns in the time series. This reflects the way performance experts visually identify the end of the warmup phase by spotting sudden drops in execution time, followed by a relatively stable phase that lasts to the end of the benchmark execution.
- Instead of evaluating global stability, we assess smoothness over sliding intervals using a probabilistic threshold derived from an adapted version of Kelly's Steady-State Detection (KSSD). This local and probabilistic evaluation of steadiness should reflect how practitioners often second-guess "borderline" cases and prefer relying on local consistency rather than global statistics.

We tailor the original approaches to the case of performance metric time series, thus introducing a novel technique that we apply to a dataset of 586 time series. We compare the results obtained with this technique to the ones obtained with Change-Point Steady State Detection (CP-SSD) [7], applied *t*-test [8] and Slope Detection Method (SDM) [9], which we consider representative targets as state-of-the-art techniques in different domains.

For the sake of comparison, we have built a ground truth on the considered dataset using judgment aggregation. With that approach, 407 series were considered steady and further evaluated for the steady-state starting indices. This approach is described in detail later on.

In this paper, we aim to address the following research questions.

RQ1. How does the new technique compare to existing ones in terms of the accuracy of steady state detection?

This is a comparison against the benchmark of the old and new methods from two points of view: a) how good the methods are at classifying time series as steady or non-steady; b) how distant (in terms of iterations and/or time) the two methods are from the ground truth.

RQ2. How does the new technique effectiveness vary while modifying its main parameters? The analysis of the model's sensitivity w.r.t. the change in its parameters. The considered parameters are the size of the sliding window for "smoothing" the outliers, the size of the sliding window checking steadiness in KB-KSSD , the *t*-test critical value used in KB-KSSD , the size of the sliding window for kernel-based step detection and the probability threshold for KB-KSSD .

The presented method demonstrates superior performance compared to CP-SSD in several key aspects. When evaluated against a ground truth set of 586 time series, KB-KSSD achieved a slightly higher number of agreements, showing a lower rate of false negatives (0.86 %) compared to CP-SSD (4.6 %). Further analysis of SSD accuracy revealed that KB-KSSD 's absolute error was 14.5 % lower than that of CP-SSD . Additionally, KB-KSSD displayed more stable behavior, with a significantly lower standard deviation of signed errors.

The paper is organized into five sections following the introduction. Section 2 defines key concepts such as *microbenchmarking*, *warm-up phase*, and *steady state*, which lay the groundwork for the subsequent analysis. Section 3 presents a detailed explanation of a novel kernel-based approach for detecting the steady state, offering insights into its methodology and underlying principles. The discussion of experimental results takes place in Section 4, where we not only present our findings but also explain the process of constructing the ground truth set used to validate our approach. Section 5 addresses potential threats to the validity of our results, suggesting possible strategies for overcoming or mitigating these concerns in future work. Finally, the paper concludes with Section 6, where a summary of the key results and insights, highlights where KB-KSSD outperforms CP-SSD method, and offers directions for further research.

The code can be accessed on the public GitHub repository,[1] with the relevant scripts released as v1.1.0.[2] This release was also published on Zenodo [10].

2. Background and related work

2.1. Java microbenchmarking

The Java Microbenchmark Harness (JMH) is the standard framework for writing and executing microbenchmarks for Java software. It helps developers create and run microbenchmarks that measure the performance of Java code segments, such as methods.

To tune the reliability of measurement, JMH supports three levels of repetitions: forks, iterations, and invocations. Invocations, the most fine-grained level of repetition, involve running microbenchmark executions continuously within a predefined time period, referred to as an iteration. A fork encompasses a series of iterations and is executed on a completely fresh Java Virtual Machine (JVM).

Within a fork, there are two types of iterations: warmup and measurement. A warmup iteration is designed to prepare the JVM for actual execution, and it ensures that measurements are not taken when the JVM is in a cold state (i.e., when it is still performing internal processes, such as class loading). In contrast, a measurement iteration is where JMH collects performance metrics, such as average execution time, for the microbenchmark.

To mitigate the effects of confounding factors, iterations should be repeated multiple times on fresh JVMs [7,11–13].

2.2. Warmup and steady state

In the first phase of their execution, Java microbenchmarks are slowly executed by the JVM. Over time, the JVM identifies frequently executed loops or methods and dynamically compiles them into optimized machine code. This process typically speeds up subsequent microbenchmark executions. Once a dynamic compilation concludes, the JVM is considered warmed up, and the benchmark is said to execute in a *steady state of performance*.

The primary goal of Java microbenchmarking is to evaluate the steady-state performance of Java applications. A common method to achieve this involves running the benchmark multiple times and discarding the first executions, which correspond to the warmup phase, to avoid skewing results. However, using a fixed number of executions does not guarantee that the warmup phase has fully ended. To address

[1] https://github.com/MartinBeseda/steady-state.git
[2] https://github.com/MartinBeseda/steady-state/releases/tag/v1.1.0

this, researchers have developed data-driven approaches to rigorously determine the end of the warmup phase. Notable methodologies include those by Georges et al. [11] and Kalibera and Jones [12]. Georges et al.'s approach relies on preset thresholds for the coefficient of variation to detect the end of warmup, while Kalibera and Jones use data visualization techniques (e.g., autocorrelation function plots, lag plots, and run-sequence plots). Despite their utility, both methods have drawbacks: Georges et al.'s heuristic often fails to reliably identify the end of warmup [12], while Kalibera and Jones' visualization-based approach is primarily manual, leading to significant limitations such as: (i) susceptibility to human error or disagreement, and (ii) lack of automation, which hampers scalability.

To address these challenges, Barrett et al. [13] introduced an automated method leveraging change point detection [14]. This technique offers a more rigorous alternative to Georges et al.'s heuristic while enabling full automation, unlike Kalibera and Jones' manual approach. The method employs a standard change point detection algorithm, PELT [15], to identify changes in the execution time of the benchmark. These detected changes are post-processed to determine if and when a benchmark reaches steady-state performance. To date, this approach represents the state-of-the-art for SSD.

3. Steady state detection using a probabilistic approach

In this section, we describe the way time series are pre-processed, together with the combination of methods used to detect (un)steadiness, and the way we evaluate the model sensitivity w.r.t. its parameters' changes.

3.1. Smoothing approach

Time series data are often noisy and contain significant outliers, which can obscure meaningful trends. To focus on the underlying trends and prevent misinterpretation, especially in cases where small oscillations may falsely suggest the conclusion of a warm-up phase, it is essential to smooth the data. This smoothing process helps mitigate the influence of high-frequency noise and enables a clearer analysis of the overall behavior.

Outlier detection in the KB-KSSD is performed by dividing the time series into several subsets. For each subset, the median is calculated as a representative value. The individual data points within each subset are then compared against specified upper and lower percentiles to identify outliers. If any data point falls outside the designated percentile range, then it is considered an outlier.

When an outlier is detected, it is replaced by the median of its corresponding subset. This replacement effectively preserves the general trend of the time series while eliminating the influence of anomalous values. By employing this technique, the time series can be effectively smoothed, allowing for more accurate trend analysis and reducing the risk of drawing false conclusions from erroneous data points.

3.2. Detection of a warm-up phase

After obtaining the smooth data, the discrete convolution given by

$$(a * t)_n = \lim_{i=\infty} \sum_{m=-i}^{i} a_i * t_{n-i} \tag{1}$$

is applied, with t and a denoting the time series we are detecting the steadiness of, and the applied convolution kernel, respectively. Two different kernels are used, both based on

$$a_i = \begin{cases} 1, & 1 \le i \le \frac{n_k}{2} \\ -1, & \frac{n_k}{2} + 1 \le i \le n_k, \end{cases} \tag{2}$$

where a_i denotes the i-th element of the kernel and n_k denotes the kernel size.

A first "large" kernel a_l is selected to cover the whole time series of length n_t, i.e., $n_t = n_k$, while the other "short" kernel a_s is of fixed length, short enough to be able to account for steps located at tails of an investigated time series. For our experimentation, a length of 15 samples was heuristically chosen, because in our context it strikes a balance between being small enough to capture localized changes, such as steps in the tails of the time series, and large enough to avoid being overly sensitive to noise. A smaller kernel is effective at detecting high-frequency features like sudden peaks or jumps, which are characteristic of steps in the tails. However, excessively small kernels may not provide sufficient context, potentially leading to misdetections [16]. Once the kernel size is reduced beyond a certain point, further decreases likely offer diminishing returns, as the kernel continues to capture the key features without losing accuracy. Furthermore, studies suggest that smaller kernel sizes are often preferable for time series analysis, as they focus on local features and reduce computational complexity [17].

After the convolution, the minima of both the resulting sequences $(a_l * t)$ and $(a_s * t)$ are located, thus detecting a step-down, if present.

After localization of both steps, it is decided which one of them, if any, is significant enough, based on the median comparison with its surroundings. The process is explained via the combination of an activity diagram and a pseudocode in the appendix.

3.3. Kelly's steady-state detection

After the potential detection of the warm-up phase termination, as it will be described in Section 3.2, the subsequent analysis of the time series is performed using Kelly's Steady State Detection (KSSD) approach [6]. Originally developed for the real-time monitoring of chemical reactors, this method enables the identification of steady-state conditions, which are characterized by minimal fluctuation in system parameters. Kelly's approach is particularly advantageous as it not only facilitates the retrospective analysis of completed time series but also supports ongoing, dynamic monitoring of the system's behavior in an "online" fashion. This capability allows for continuous assessment of whether the system has stabilized or is undergoing transient changes, providing a valuable tool for real-time process control.

In summary, the KSSD algorithm combines statistical methods and mathematical expressions to effectively detect steady states in continuous processes, ensuring that the process is operating within acceptable limits for optimization and control.

The first assumption of the KSSD algorithm is that the time series behavior is operating with a non-zero slope multiplied by its relative time within the window given by

$$x_t = mt + \mu a_t, \tag{3}$$

where mt denotes the deterministic drift component; μ denotes the mean of a hypothetical stationary process equaling the sample mean over the time window with zero slope; a_t denotes the independent and identically distributed random error series with zero mean and a standard deviation σ_a. The index t denotes the iteration at which the sample is collected, thus representing a so-called "random walk with drift" [18]. Without going into in-depth mathematical reasoning behind the method, it can be seen that the mean of x_t denoted as μ is defined as:

$$\mu = \frac{1}{n}\left(\sum_{t=1}^{n} x^t - m\sum_{i=1}^{n} t\right) \tag{4}$$

and that the expected value of $a_t - a_{t-1}$ is zero with a standard deviation of $2\sigma_a$, which can be estimated as

$$\sigma_a = \sqrt{\frac{1}{n-2}\sum_{t=1}^{n}(x_t - mt - \mu)^2}. \tag{5}$$

Now, together with a specified t-test threshold value t_{crit} at a specific significance level, everything necessary is available to test the null hypothesis that the time series values are steady around μ. This hypothesis

can be formalized as:

$$\begin{cases} y_t = 1, & |x_t - \mu| \leq t_{\text{crit}}\sigma_a \\ y_t = 0. \end{cases} \tag{6}$$

The likelihood that the null hypothesis is false can be expressed by

$$\mathcal{L}(y) = \frac{1}{n}\sum_{t=1}^{n} y_t, \tag{7}$$

as it is the fraction of time where the time series is considered to be steady.

3.4. Parameter sensitivity analysis

The sensitivity analysis of KB-KSSD algorithm is performed via Sobol's indices [19], as the model's response is not generally linear and one-at-a-time analysis is not sufficient due to the presence of significant interactions of higher orders w.r.t. the model parameters.

Sobol's indices are used in global sensitivity analysis to quantify the contribution of each input variable to the output variance of a mathematical model. Given a model f with input variables $\mathbf{X} = (X_1, X_2, \ldots, X_d)$ and output $Y = f(\mathbf{X})$, the total variance of the output Y is given by

$$\text{Var}(Y) = \text{Var}(f(\mathbf{X})), \tag{8}$$

where

$$f(\mathbf{X}) = \sqrt{\textstyle\sum_{i=1}^{d} |X_i|^2} \tag{9}$$

represents L2-norm, i.e., the sum of squared errors cost function, which quantifies the deviation from a manually-selected ground truth.

Sobol's first-order sensitivity indices S_i measure the contribution of each individual input variable X_i to the total variance of the model output. Mathematically, the first-order index for X_i is defined as the ratio of the variance of the output conditioned on X_i to the total variance of the output given by

$$S_i = \frac{\text{Var}_{X_i}(f(\mathbf{X}))}{\text{Var}(f(\mathbf{X}))}, \tag{10}$$

where $\text{Var}_{X_i}(f(\mathbf{X}))$ represents the variance of $f(\mathbf{X})$ when all inputs except X_i are held constant.

In addition to first-order indices, Sobol's method includes higher-order sensitivity indices, which measure the contribution of interactions between input variables to the total variance. For example, the second-order index S_{ij} represents the contribution of the interaction between variables X_i and X_j, and is given by

$$S_{ij} = \frac{\text{Var}_{X_i,X_j}(f(\mathbf{X}))}{\text{Var}(f(\mathbf{X}))}, \tag{11}$$

where $\text{Var}_{X_i,X_j}(f(\mathbf{X}))$ denotes the variance of the output due to the interaction between X_i and X_j. Additionally, Sobol's total sensitivity indices S_i^{tot} measure the total effect of X_i, which includes both the direct effect of X_i and its interactions with other inputs. The total index is defined as:

$$S_i^{\text{tot}} = 1 - \frac{\text{Var}_{X_i^{\perp}}(f(\mathbf{X}))}{\text{Var}(f(\mathbf{X}))}, \tag{12}$$

where $X_i^{\perp}$ represents all input variables excluding X_i. This total index captures the influence of X_i on the output, both directly and through interactions with other variables.

In this paper, SALib implementation[3] of Sobol's sensitivity analysis was used.

3.5. Model fitting

The model optimal configuration was approximated via a grid search over 15,000 different parameter configurations w.r.t. the Manhattan distance

$$m(\mathbf{X}|\theta) = \sum_{i=1}^{n} |p(\mathbf{X}|\theta) - g(\mathbf{X}|\theta)|, \tag{13}$$

with p and g denoting the model's prediction and the ground-truth value corresponding to the vector of input variables $\mathbf{X}$ representing the time series included in the ground truth and the selected set of parameters θ. The time series evaluated as unsteady by KB-KSSD were not considered in the "training" process. The process itself was straightforward, selecting the set of parameters $\tilde{\theta}$, for which the Manhattan distance from the ground truth is minimal, as given by

$$\min_{\theta} m(\mathbf{X}|\theta). \tag{14}$$

4. Results

The reference set that represents our ground truth was obtained in three phases: (i) random selection of 586 of time series from a dataset[4], (ii) binary classification of (un)steadiness of every time series, and (iii) manual selection of index where the steady state starts in time series judged as steady. The binary classification in step (ii) was performed via *judgment aggregation* [20–22] in a group of five researchers. The steady-state index was then selected individually by every researcher, and the five indices were checked for clusters with DBSCAN method [23]. If there were three or five indices clustered, the middle point was selected as a reference steady state index. We did the same where all indices were scattered, i.e., where the "judges" disagreed more heavily. Finally, if there were four indices clustered, the third of them is taken as a reference point, as the most conservative judgment. We remark that, besides other analyses, we have also separately analyzed the cases of agreement and disagreement. From now on, we will refer to this set as Judgment-Aggregated Ground Truth (JAGT).

The objective of this section is to address the research questions introduced in Section 1, by conducting a comprehensive evaluation of KB-KSSD algorithm in two key aspects. First, we perform a comparative analysis between KB-KSSD , CP-SSD [7], SDM [9], and the applied *t*-test [8], focusing on their respective accuracy and reliability to detect steady state. The in-depth comparison is being performed with respect to CP-SSD method, as it was found to be the most effective one from the selected reference methods. Second, we investigate the sensitivity of KB-KSSD to changes in its key parameters, aiming to quantify how changes in these parameters influence its overall performance and robustness. This dual approach provides a thorough understanding of both the comparative efficacy and the parameter sensitivity of KB-KSSD .

4.1. RQ1. How does KB-KSSD compare to existing methods in terms of the accuracy of steady-state detection?

In this section, we describe a thorough numerical analysis of different aspects of KB-KSSD , providing both an in-depth comparison with the previous S-o-A approach.

The optimal configuration, determined through this grid search described in Section 3.5, consists of a window size of 100 samples for outlier removal, a window size of 500 samples to assess steadiness as per KB-KSSD , a critical *t*-test value of 4.0 within Kelly's steadiness evaluation, a window size of 70 samples for step detection, and a probability threshold of 0.95 for determining steadiness according to Kelly's part again.

a) *How good are the methods at classifying time series as steady or non-steady?*

[3] https://salib.readthedocs.io/en/latest/

[4] https://github.com/SEALABQualityGroup/steady-state

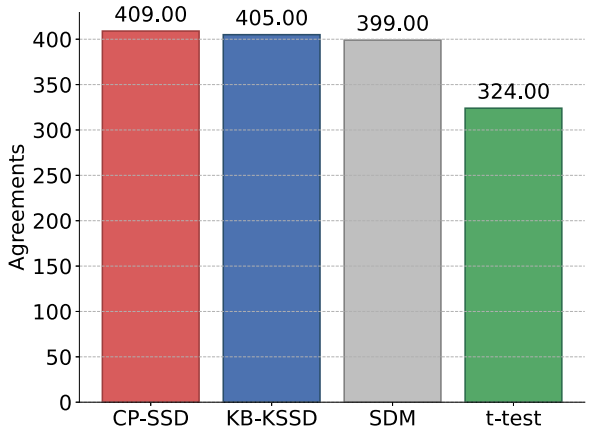


Fig. 2. Comparison of successful binary steadiness classifications.

Firstly, we compare all the methods with respect to their ability of binary (un)steadiness classification on the full JAGT, as illustrated in Fig. 2. It can be seen that, while CP-SSD , KB-KSSD and SDM perform almost identically well with 409, 405 and 399 agreements with JAGT, respectively, the applied *t*-test performs significantly worse with its 324 agreements, thus only performing at 80 % of KB-KSSD efficiency.

Moreover, the Manhattan distance given by Eq. (13) was used to further evaluate the ability of KB-KSSD compared to the reference methods with respect to the predicted distance from the steady-state indices in JAGT. These results are illustrated in Fig. 3. In this case, KB-KSSD performed the best, with a total distance of 81519, compared to the distances of 93047, 95,958 and 203,237 of CP-SSD , SDM and the *t*-test, respectively. Furthermore, the steady timeseries were also considered separately, where the judgment-aggregated indices were clustered and scattered, respectively. In both cases KB-KSSD performed better with the distances being 35,698 / 45,821 on timeseries with clustered and scattered indices, respectively. These distances notably differ from the other compared methods with distances of 39,438 / 53609, 50,437 / 48,521 and 106,703 / 96,534 signifying clustered/scattered indices for CP-SSD , SDM and the *t*-test, respectively. It is interesting to note that both SDM and the *t*-test did not perform better on the timeseries with clustered indices, probably due to their general nature of techniques not developed for performance timeseries analysis.

We deduce that the main reason behind KB-KSSD superior performance is the kernel-based detection of “downward steps” denoting the end of a warm-up phase. In addition, the usage of two different kernels provides the flexibility of detecting such steps even near the borders of a timeseries, thus overcoming typical disadvantages in the application of a single convolution kernel. To illustrate this observation, three different timeseries can be seen in Fig. 4. The details of downward steps, convolutions, and the median-based decision of step significance are illustrated on the corresponding timeseries in Fig. 5, where the detected steady state index corresponds to the minimal value of the (yellow) convolution curve.

Given that SDM and the applied *t*-test performed consistently worse than KB-KSSD and CP-SSD , we continued only with statistical analysis of the latter two methods.

In Fig. 6 we present the number of agreements and disagreements regarding the (un)steadiness classification for both KB-KSSD, CP-SSD, and JAGT, including manually-selected unsteady time series. There were 586 time series considered in the “full” ground truth set, 407 of them were labeled as steady. Both KB-KSSD and CP-SSD were evaluated on all the time series, and it was found that both methods performed very similarly, achieving 409 and 405 agreements, i.e., ~69.8 % and ~69.1 %, w.r.t. CP-SSD and KB-KSSD , respectively. KB-KSSD classified 176 (~30 %) as steady, while they were not, as compared to 150 (~26 %) false positives obtained via CP-SSD . For the false negatives (i.e., classifying as unsteady a time series that was labeled as steady in the ground truth), KB-KSSD misclassified 5 (~0.86 %), while CP-SSD 27 (~4.6 %). Both methods simultaneously agreed with the ground truth in 379 (65.7 %) cases. Both methods were found to agree with each other and to disagree with the ground truth at the same time in 151 (25.8 %) cases.

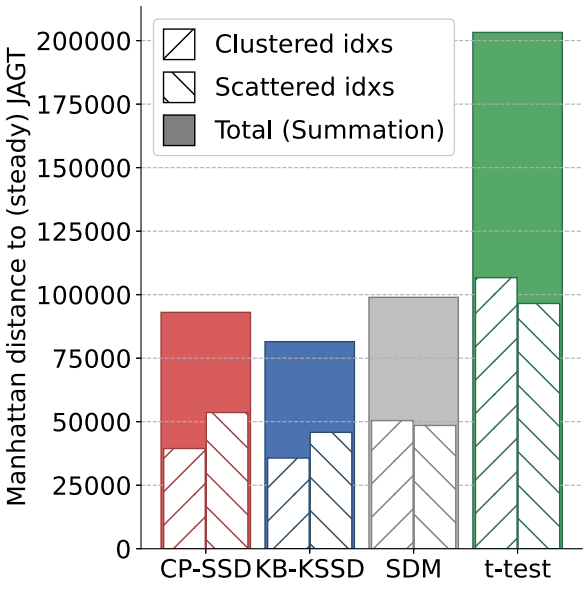


Fig. 3. Comparison of prediction errors w.r.t. the steady-state indices in JAGT.

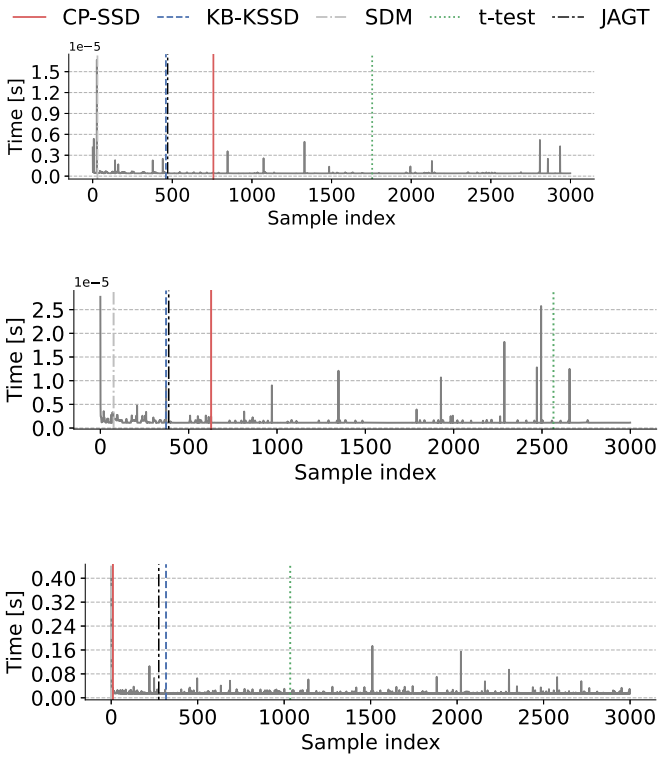


Fig. 4. Three different timeseries with different steady-state indices detected via the considered methods, w.r.t. JAGT.

Furthermore, the number of false positives was partitioned w.r.t. the character of the ground truth labels, which can be: (i) clustered, in case of a strong agreement among human classifiers, or (ii) scattered, in case of some disagreement among human classifiers, consistent with the cases mentioned at the beginning of this section. In Fig. 7, it is shown

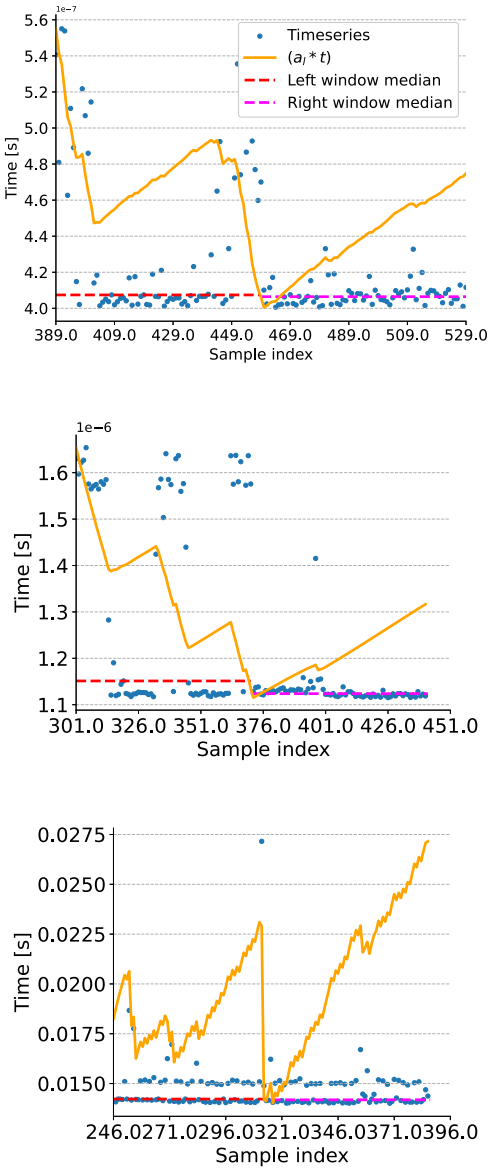


Fig. 5. Details of the steady-state detection (i.e., downward steps, convolutions, and the median-based decision of step significance) of the timeseries in Fig. 4.

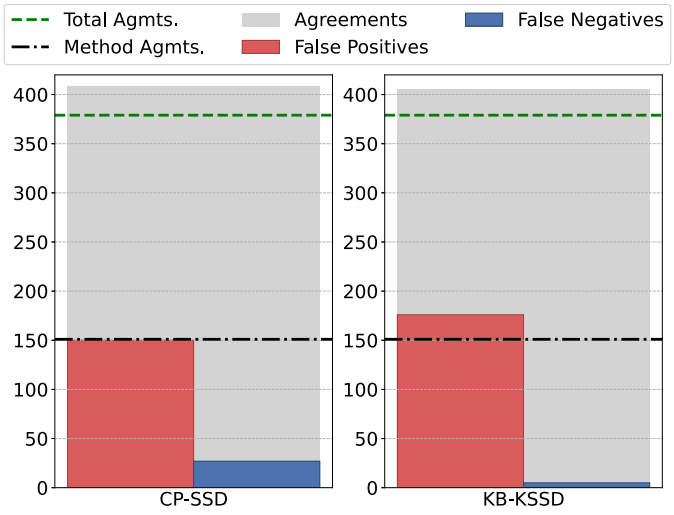


Fig. 6. Number of (dis)agreements w.r.t. the larger dataset including manually-detected unsteady series. The number of total agreements denotes situations where both methods and the ground truth agree with each other, while the number of method agreements denotes the situation where both methods agree with each other, but not with the ground truth.

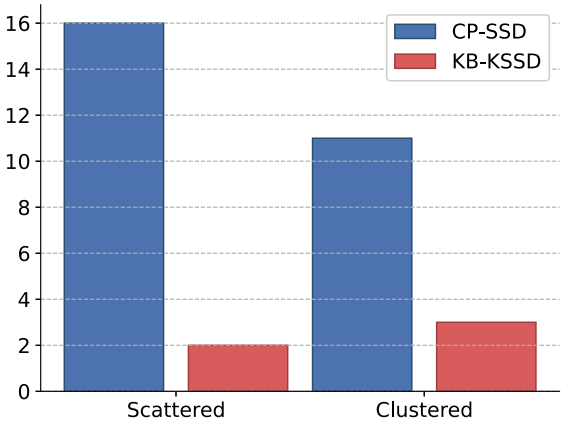


Fig. 7. Number of detected unsteady time series by both approaches w.r.t. the ground truth labels.

that CP-SSD delivered falsely negative results in 16 and 11 cases for time series with scattered and clustered ground truth steadiness indices, respectively, while KB-KSSD provided 2 and 3 of them.

b) *How distant (in terms of iterations and/or time) the two methods are from the ground truth?*

For obvious reasons, the following investigation on the ability to correctly find the index of steady-state beginnings is only based on the set of steady time series as reference ground truth.

Consequently, we analyzed both the raw differences from the ground truth and their absolute values to correctly assess a possible directional bias in model results, KB-KSSD reliability, and accuracy. All the statistical tests were performed with a 0.05 level of significance.

By analyzing the raw errors (i.e., the differences between the ground truth and the value identified via KB-KSSD) visualized in Figs. 8 and 9, we calculated the mean raw errors and the corresponding standard deviations for KB-KSSD and CP-SSD as $\mu_N = 255.7$, $\sigma_N = 522.3$ and $\mu_R = 110.3$, $\sigma_R = 658.40$, respectively. Based on these findings we can assert that 95 % of all errors fall within the intervals $\langle -789.4; 1299.6 \rangle$ and $\langle -1206.5; 1427.1 \rangle$ for KB-KSSD and CP-SSD , respectively. As it can be seen, the 95 % interval is longer by ~26.1 % in the case of CP-SSD . Also, the mean raw errors of both methods are larger than zero, thus implying a directional bias in both of them. Since KB-KSSD delivers a higher mean, we tested whether the difference in bias is statistically significant. Hence, we checked the differences between the sets of errors, shown in Fig. 10, to assess their normality and symmetry, thus testing assumptions for *t*-test [24] and Wilcoxon signed-rank test [25]. As the Shapiro-Wilk test [26] resulted in a statistic ~0.72 and a p-value 0, it can be safely concluded that the distribution of differences is not normal, rendering the *t*-test unfit for making the decision. For the symmetry assessment, we computed the skewness and we obtained a -1.14 value, which signifies a moderate skewness to the left, rendering the Wilcoxon test also unreliable [27]. With this in mind, we settled for a sign test, thus observing that the difference in directional bias is significant, i.e.,

Both methods are similarly efficient when it comes to the binary classification of (un)steadiness. Even if KB-KSSD tends to classify a time series incorrectly as steady more often than CP-SSD , KB-KSSD is more reliable with respect to false negatives, *i.e.,* classifying the series as unsteady, while it is steady.

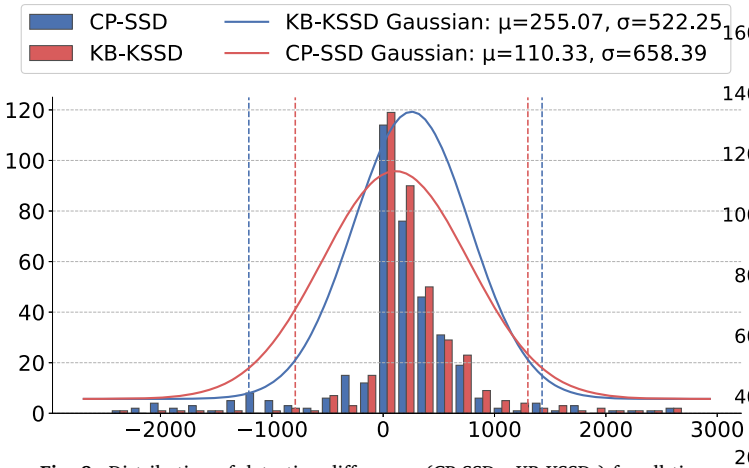


Fig. 8. Distribution of detection differences (CP-SSD - KB-KSSD) for all time series. The plot also visualizes the range, where 95 % of all errors stay within the vertical lines, and there are Gaussians included to visualize an approximate normal distribution with the same μ and σ as in the real error distributions for easier readability.

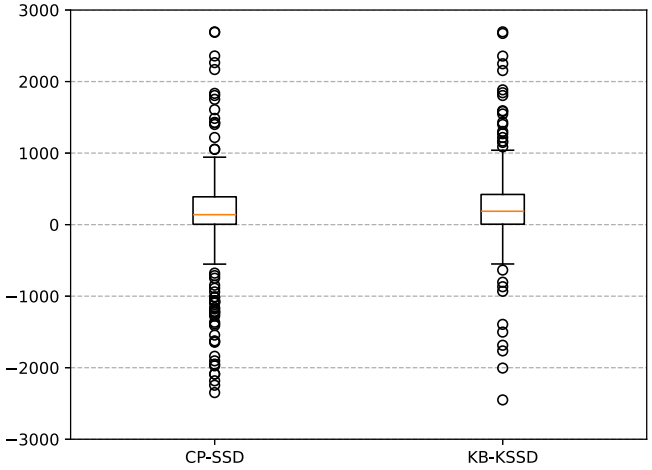


Fig. 9. Detection differences (CP-SSD - KB-KSSD) for all time series.

KB-KSSD tends to "underestimate" the position of the steady state more than CP-SSD . Albeit not so large considering the whole difference range, this remains one of the KB-KSSD properties, which needs to be further addressed in future work. To be sure that the models' errors do not follow the same distribution, we checked it with the Cramer-von Mises test [28] to uniformly evaluate the differences, the Kolmogorov-Smirnov test [29] to evaluate the differences emphasizing the behavior around the expected value, and the Anderson-Darling test [30], mainly focused on distribution tails. All the tests agreed that the raw error distributions are different.

Thereafter, we intended to assess the models' reliability, i.e., the ability to provide reasonable results without many significant outliers. In particular, we wanted to test if the difference in the distributions' variances is significant, or if the previous results were mostly influenced by the difference in the expected values. For this goal, we utilized the Levene test [31,32], which has rejected the possibility of similar variances, thus confirming that KB-KSSD provides more stable behavior.

To mitigate the possible problem with false discoveries, i.e., falsely rejecting the null hypothesis, we applied Benjamini-Hochberg correction [33]. All results of these tests, i.e., the significance of differences in both expectation values and variances, can be seen in detail in Table 1.

The accuracy of both models was established based on their Manhattan distance, given by Eq. (13), from the ground truth. As visualized in Fig. 11(a) and (b), we found out that KB-KSSD performed with a total error of 137094, while CP-SSD performed with 160323, i.e., with an error larger by ∼14.5 % than the KB-KSSD one. Considering the accuracy for the clustered and scattered indices in the ground truth separately, KB-KSSD performs with the errors of 61,924 and 75170, respectively, while CP-SSD performs with 70,579 and 89744. It can be seen that, as expected, while both methods perform better when the indices were clustered (i.e., when the steady state was identified by human classifiers in agreement), KB-KSSD is more accurate in both cases, with CP-SSD exceeding its errors by ∼12.3 % and 16.2 %. About the expected performance, the mean absolute values and the standard deviations are $\mu_{NA} = 363.6$, $\sigma_{NA} = 453.4$ and $\mu_{RA} = 425.3$, $\sigma_{NA} = 514.6$ for KB-KSSD and CP-SSD , respectively. Thus, considering the error magnitudes, CP-SSD performs with an error higher by ∼14.5 % on average, while the standard deviation of its absolute errors is higher by ∼11.9 %, thus signifying more frequent errors by large magnitude.

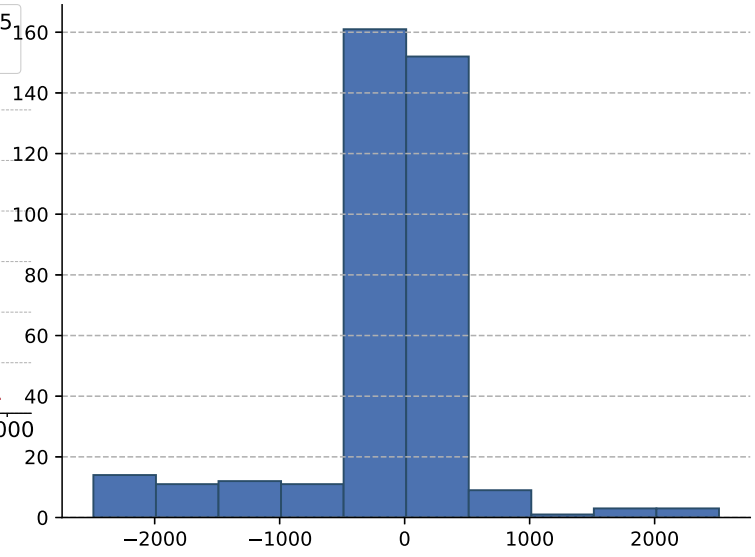


Fig. 10. Differences between paired raw errors.

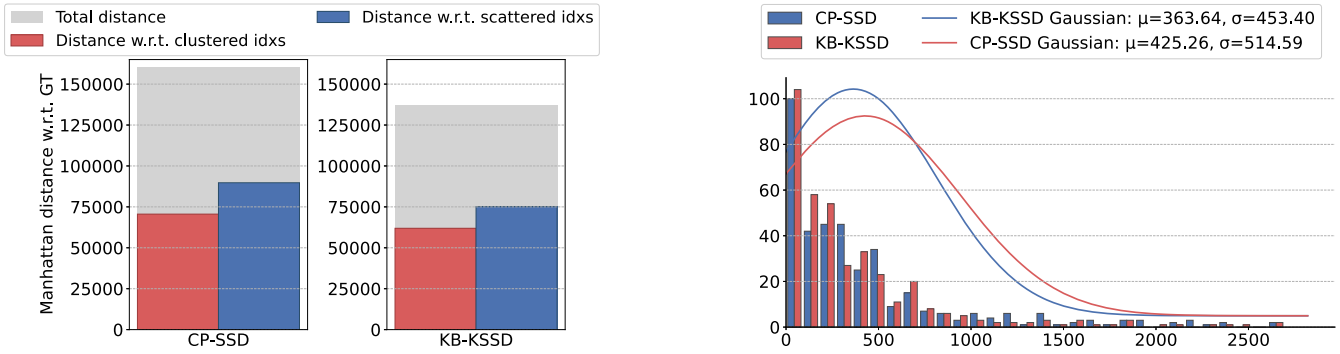
Table 1
Statistical tests comparing the raw error distributions.

Test	Statistic	p-value	BH-adjusted p-value
Sign	0.58	0	0.01749295
CvM	0.51	0.04	0.02313882
KS	0.11	0.02	0.02150996
AD	3.61	0.01	0.03670551
Levene	6.21	0.01	0.02150996

To further scrutinize these observations and to generalize them to the assumed population instead of only the limited set of observations, we checked again whether the differences in the populations' expected values were significant.

For this, we needed to select a convenient statistical test again. The differences between absolute errors, visualized in Fig. 12, are not normal in the sense of Shapiro-Wilk, as shown by statistic ∼0.73 and a p-value 0. Thus, we need to discard the *t*-test. Moreover, the differences are not even symmetric, as the skewness equal to ∼0.59 suggests that the differences are skewed to the right rather significantly, thus rendering also the Wilcoxon signed-rank test unreliable. The finally adopted signed rank test eventually suggested that neither of these absolute error distributions possesses a similar expected value with its statistic 0.67 and a p-value almost equal to 0.

The comparative analysis has shown that KB-KSSD consistently outperforms CP-SSD in several critical areas. KB-KSSD demonstrated a lower false negative rate, classifying significantly fewer unsteady time series as steady. This shows that KB-KSSD is more sensitive to unsteady periods, an essential feature for accurate time series classification. When



(a) Bar plot of Manhattan distances from the ground truth of CP-SSD and KB-KSSD

(b) Histogram of Manhattan distances from the ground truth.

Fig. 11. Manhattan distances from the ground truth.

KB-KSSD outperforms CP-SSD in terms of accuracy (error reduction by 14.5 %), reliability (fewer extreme outliers), and sensitivity (lower false negative rate). It provides more precise and stable results, with narrower error margins and a more reliable performance across different time series.

detecting the onset of steady states, KB-KSSD provided more precise results, with narrower error margins and more stable performance. While both methods exhibited some directional bias, KB-KSSD showed fewer extreme outliers, further supported by the fact that KB-KSSD had a narrower 95 % confidence interval for detection errors, indicating more reliable predictions. In terms of accuracy, KB-KSSD reduced the total error (measured by Manhattan distance) by approximately 14.5 % compared to CP-SSD . This performance improvement was evident in both well-clustered and more scattered ground truth data. These findings suggest that KB-KSSD is not only more accurate in general, but also more capable of handling challenging or ambiguous scenarios, making it a better fit for real-world applications. Its reduced error magnitude and higher precision across different time series make it a more reliable and efficient tool for SSD.

4.2. RQ2. How does KB-KSSD effectiveness vary while modifying its main parameters?

To address RQ2, KB-KSSD was analyzed using Sobol's indices described in Section 3.4, thus assessing its sensitivity w.r.t. changes in its parameters. As parameters to be investigated, we chose the size of the window used in KSSD (`prob_win_size` $\in \langle 400; 600 \rangle$), the size of the window for step-detection (`step_win_size` $\in \langle 50; 90 \rangle$), the critical value for the *t*-test included in KSSD (`t_crit` $\in \langle 3; 5 \rangle$), the probability threshold for steadiness acceptance (`prob_threshold` $\in \langle 0.75; 0.95 \rangle$), and the size of the window used for detecting outliers (`outlier_win_size` $\in \langle 80; 120 \rangle$). The investigated intervals were selected around the "optimal" configuration found by the grid search described in Section 3.5. Moreover, there were 24,576 samples generated for the analysis.

These parameters were selected so as to capture most of the degrees of freedom, i.e., to represent most of the flexibility of KB-KSSD . Other parameters, like percentiles in the outlier removal phase described in Section 3.1, were not included, as they are not a core part of the method. Also, the size of the lesser kernel was not investigated in this work, as it is usually enough to select a small enough kernel to detect patterns in series tails, and from that point on, the behavior does not change.

In this analysis, indices S_1, S_2, and S_T denoting different sensitivity measures were considered. S_1 (first-order Sobol index) quantifies the direct effect of an input variable on the output variance, S_2 (second-order Sobol index) measures the contribution to the output of interactions between pairs of variables, and S_T (total Sobol index) captures the total effect of an input, including its main effect and interactions with all other variables. S_T is generally larger than or equal to S_1 as it accounts for both individual and interaction effects. In Table 2 it can be seen that the most significant parameters w.r.t. their changes are `prob_win_size` and `t_crit`, followed by `prob_threshold`, `outlier_win_size` and `step_win_size`, while the model is almost "immune" to the change of `step_win_size` in the chosen interval. The first-order effects do maintain the influence order aside from the fact that the model is somewhat more sensitive to the change of `outlier_win_size` than to the change of `prob_threshold`. The large confidence intervals and the fact that there are some badly converged (less than 0) indices of the second-order effects indicate that there were not enough samples to assess these separately in a reliable manner. All of the S_2 values are listed in a table in the appendix. That said, we can reasonably assume that KB-KSSD is basically insensitive to changes in the interactions of `step_win_size` with `outlier_win_size`, `prob_threshold`, and `t_crit` due to their indices being ~ 0, with confidence intervals (-0.01, 0.02), (-0.01, 0.01), and (-0.01, 0.01), respectively. It can be seen that some negative values appear in the confidence intervals, which are due to a lack of computational convergence. However, the estimations can still be considered reliable, as not only are the estimated indices almost zero, but the obtained intervals are also very narrow. It can be further assumed that the only significant interaction exists between `prob_threshold` and `t_crit`, as its S_2 index is ~ 0.13, i.e., the only one significantly higher than 0, and its confidence interval is (0.04, 0.23), i.e., the only one being fully above than 0. Still, an analysis with a larger number of samples could allow for deducing some further information from S_2 indices, but it implies very high computation demands.

Table 2
Sobol's indices of the first order S_1 and the total ones S_T together with corresponding confidence intervals for the 0.95 confidence level.

	S_1	S_T
prob_win_size	0.19718534 (0.15, 0.24)	**0.46349316** (0.41, 0.51)
step_win_size	0.00631332 (-0.0, 0.01)	**0.01394031** (0.01, 0.02)
t_crit	0.15670927 (0.11, 0.2)	**0.46206506** (0.41, 0.52)
prob_threshold	0.1107751 (0.07, 0.15)	**0.35747568** (0.31, 0.41)
outlier_win_size	0.15383209 (0.12, 0.19)	**0.26101527** (0.24, 0.28)

S_1, S_2 and S_T are visualized together with their corresponding confidence intervals in Fig. 13.

To assess the stability of S_T estimation, another analysis with 6144 samples was performed, with the corresponding S_T values listed in Table 3. In this case, the order of parameters obtained by considering the S_T sensitivity is almost the same, with the model seemingly more sensitive to the change of `t_crit` than to the change of `prob_win_size`. Talking about the confidence intervals, the analysis with a larger num-

Table 3
Sobol's S_T indices for the analysis performed with only 6144 samples and the S_T percentage w.r.t. the S_T obtained via 24,576 samples (in Table 2). Performed at 0.95 significance level.

Parameters	S_T	δ
prob_win_size	0.31653608 (0.26, 0.38)	**68.3 %**
step_win_size	0.01159328 (0.01, 0.01)	**83.2 %**
t_crit	0.39023183 (0.31, 0.47)	**84.5 %**
prob_threshold	0.2987368 (0.22, 0.37)	**83.6 %**
outlier_win_size	0.19648237 (0.16, 0.23)	**75.3 %**

KB-KSSD is robust to variations in key parameters, with its performance largely determined by a few influential factors. The size of the window used for steadiness evaluation and the critical value for the *t*-test had the most significant impact on performance.

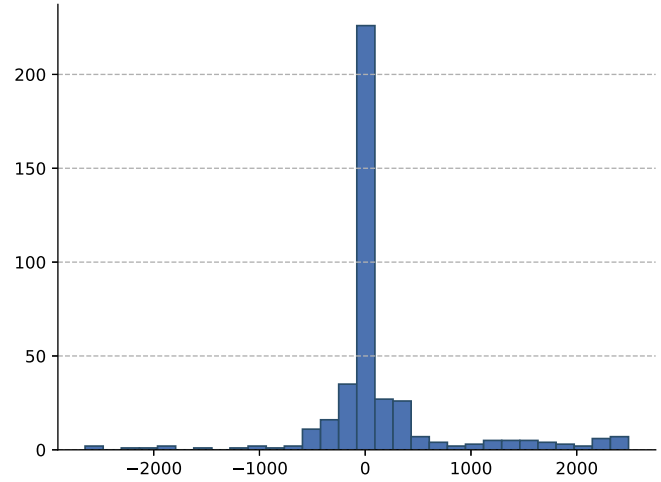


Fig. 12. Histogram of differences of Manhattan distances from the ground truth.

ber of samples provides intervals with a mean ∼0.071 and the standard deviation of ∼0.04, while the analysis using a smaller number of samples provides intervals with mean of ∼0.102 and a standard deviation of ∼0.059, i.e., larger by ∼44.4 % and ∼47.5 %, respectively.

The sensitivity analysis of KB-KSSD was conducted using Sobol's indices, focusing on key parameters such as window sizes and thresholds. It showed that KB-KSSD is robust to variations in key parameters, with its performance largely determined by a few influential factors. Among the parameters studied, the size of the window used for steadiness evaluation (`prob_win_size`) and the critical value for the *t*-test (`t_crit`) had the most significant impact on performance, while other parameters like `step_win_size` have minimal impact. Despite this sensitivity, the method maintained stable and reliable results across a range of parameter settings, indicating that it can be effectively tuned without risking large drops in performance.

5. Threats to validity

Construct validity. We assume that benchmarks reaching a steady state will do so within the defined execution time. Some benchmarks may require more time to stabilize. However, our selected dataset was constructed by running benchmarks significantly longer than usual (each benchmark was run at least 3 000 times across 10 forks) [7], exceeding typical developer practices and experiments in previous literature [13].

The process of establishing the ground truth relies on human judgment for both binary classification and steady-state index selection, which introduces potential subjectivity and inconsistency. While judgment aggregation was employed to mitigate individual biases, as done

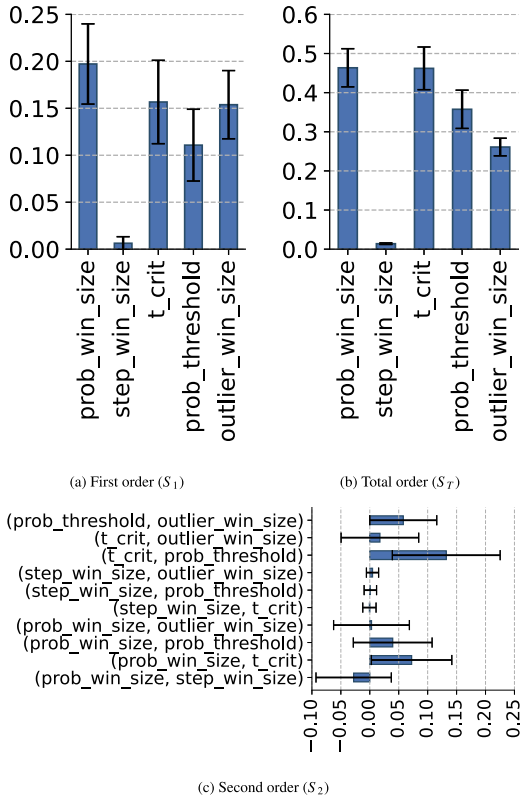


Fig. 13. Sobol's indices of the different selected parameters denoting the model's sensitivity w.r.t. them. The plot describes the indices of the 1st order S_1, 2nd order S_2, and the "total" index S_T denoting the estimated behavior with all the higher order interactions involved. Also, the plot contains the confidence intervals of the index estimates for the 0.95 confidence level.

in previous studies [20–22], the method does not eliminate the possibility of systematic bias among the five researchers. This might lead to a ground truth that does not fully represent the true underlying characteristics of the data, affecting the validity of subsequent analyses and conclusions.

Internal validity. The process of identifying and replacing outliers may inadvertently distort the true characteristics of the time series data. By relying on median-based filtering within predefined percentile thresholds (Section 3.1), genuine but extreme variations in benchmark performance could be misclassified as noise and removed. Nonetheless, filtering outliers in time series data is a common and essential preprocessing step to improve the accuracy of changepoint and trend detection [34].

Since only 10 % of the available time series were annotated, there is a risk of selection bias. Even though a one-time series was chosen for each benchmark to ensure representation, this subset may not fully capture the variability present in the entire dataset.

The use of a graphical tool for visual inspection and annotation introduces a potential source of inaccuracy. Since the tool approximates the selected data point to the closest one to the user's mouse click, slight misalignments may occur, leading to minor errors in the ground truth annotations.

Changes in annotating standards over time could impact consistency, as annotators may unintentionally shift their interpretation of the annotation criteria throughout the annotation process. To avoid that, as this would introduce inconsistencies in the ground truth, recurring meetings were held to share concerns and realign the group on the basis of the ongoing annotation experience.

External validity. The external validity threats of this work are primarily inherited from the dataset chosen for evaluation. Specifically, the dataset is limited to GitHub repositories, making it uncer-

tain whether the findings generalize to other open-source hosting platforms or industrial software. Additionally, the dataset focuses solely on microbenchmarking using JMH. While JMH is widely adopted, the applicability of our findings to other benchmarking practices or settings remains unknown.

Conclusion validity. KSSD assumes that the random error component of the time series is independently and identically distributed (i.i.d.). However, in real-world benchmarking scenarios, performance measurements often exhibit autocorrelation, where successive data points are not entirely independent due to ongoing optimization occurring at the JVM or operating system level. If this assumption is violated, the KSSD algorithm may misinterpret correlated variations as steady-state phases. Nevertheless, KSSD parameters, namely the sliding window size, can be tuned to mitigate these effects, allowing for some adaptation to real-world deviations from the i.i.d. assumption, as it can be seen in similar real-world applications, where we encounter "structural breaks" in series [35,36]. Also, the i.i.d. assumption could be eased or completely removed when an *autoregressive model (ARM)* would be incorporated into KSSD and the residuals from the ARM would be used in KSSD instead, as it is a quite common application in different fields dealing with discrete time series [18,37].

6. Conclusion

This study introduced and evaluated KB-KSSD for detecting steady states in time series data, and it has compared its performance to CP-SSD . The evaluation addressed two research questions by assessing both the comparative accuracy of the methods and the sensitivity of KB-KSSD to variations in key parameters.

Overall, KB-KSSD offers several advantages over CP-SSD . It is more accurate in detecting both steady and unsteady states, with a significantly lower false negative rate and reduced error magnitude. The method is also more robust, delivering reliable results even when key parameters are modified. This makes KB-KSSD highly suitable for applications requiring precise and consistent steady-state detection, especially in benchmarking scenarios where performance must be both accurate and reproducible. Its ability to handle ambiguous cases and deliver stable results across parameter variations provides added flexibility for users, making it a preferable choice compared to existing approaches.

The presented model exhibits several limitations that should be addressed in future work. Firstly, the training process utilized a simple cost function, which could be enhanced to account not only for the overall deviation from ground truth, but also for the number of false positives and false negatives, directional bias, error variance, and both mean and median absolute errors. These metrics could be weighted differently to optimize model performance. Furthermore, this multi-objective optimization would benefit from the application of a more robust optimization technique with a strong exploration component, such as particle swarm optimization or evolutionary algorithms, rather than relying on a simple grid search. However, it is important to note that these methods introduce their own challenges, including the risk of convergence to suboptimal local minima, vanishing gradients, barren plateaus (for gradient-based methods), and other issues.

The "short" convolution kernel a_s was fixed to a size of 15 based on an educated guess. This choice has not been rigorously investigated or optimized, and future work should explore different kernel sizes to assess whether a more refined approach could improve sensitivity and detection accuracy. Systematically evaluating the impact of varying kernel sizes would provide more insight into the model's robustness and effectiveness across different scenarios. Furthermore, the KB-KSSD sensitivity to changes in a_s length would be better understood via Sobol's indices.

Another limitation of the current model is its inability to detect upward steps, as it is exclusively designed to identify downward transitions in the data. This constraint limits the model's applicability, particularly in scenarios where upward shifts in the system behavior may occur, such as during steady-state analysis of software runtime. To address this, future work should focus on extending the model to capture both upward and downward steps, thereby enhancing its ability to fully characterize the dynamics of the system under various conditions.

Finally, it would be advantageous to implement more targeted preprocessing techniques tailored to the specific characteristics of the time series. This aspect is closely linked to the broader issue of time series classification, which will be the focus of future research efforts.

CRediT authorship contribution statement

Martin Beseda: Writing – original draft, Software, Methodology, Investigation; **Vittorio Cortellessa:** Writing – review & editing, Validation, Supervision, Project administration, Funding acquisition; **Daniele Di Pompeo:** Writing – review & editing, Validation, Investigation; **Luca Traini:** Writing – review & editing, Validation, Investigation; **Michele Tucci:** Writing – review & editing, Validation, Investigation.

Data availability

The full replication package is openly accessible and linked to in the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Martin Beseda was supported by Italian Government (Ministero dell'Università e della Ricerca, PRIN 2022 PNRR) – cod. P2022SELA7: "RECHARGE: monitoRing, tEsting, and CHaracterization of performAnce Regressions" – D.D. n. 1205 del 28/7/2023 . Daniele Di Pompeo and Michele Tucci were supported by European Union – NextGenerationEU – National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) – Project: "SoBigData.it – Strengthening the Italian RI for Social Mining and Big Data Analytics" – cod. IR0000013 – D.D. n. 3264 del 28/12/2021 . Luca Traini was supported by European Union – NextGenerationEU – National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) – Project: "ICSC - Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing" – cod. CN00000013 – D.D. n. 3138 del 16/12/2021 . Vittorio Cortellessa was supported by European Union – NextGenerationEU – National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) – Project: "Modeling and Analyzing Tradeoffs beTween cybERSecurity and other quality attributes in Cyber-Physical Systems" (MATTERS), in the context of Spoke 8 - Risk Management and Governance (SERICS) - "SEcurity and RIghts In the CyberSpace" (CUP: J33C22002810001).

References

[1] C. Truong, L. Oudre, N. Vayatis, Selective review of offline change point detection methods, Signal Process. 167 (2020) 107299. https://doi.org/10.1016/j.sigpro.2019.107299

[2] A. Pushkar, M. Gupta, R. Wadhvani, M. Gyanchandani, A comparative study on change-point detection methods in time series data, in: 2022 2nd International Conference on Intelligent Technologies (CONIT), 2022, pp. 1–7. https://doi.org/10.1109/CONIT55038.2022.9848051

[3] H. Xu, J. Wu, T.-L. Tseng, An efficient method for online identification of steady state for multivariate system, in: International Manufacturing Science and Engineering Conference, 51388, American Society of Mechanical Engineers, 2018, V004T03A005. https://doi.org/10.1115/MSEC2018-6565

[4] E.M. Turan, J. Jäschke, A simple two-parameter steady-state detection algorithm: concept and experimental validation, in: Computer Aided Chemical Engineering, 52, Elsevier, 2023, pp. 1765–1770. https://doi.org/10.1016/B978-0-443-15274-0.50280-8

[5] C. Nellis, T. Danielson, A. Savara, C. Hin, The Ft-Pj-RG method: an adjacent-rolling-windows based steady-state detection technique for application to kinetic Monte Carlo simulations, Comput. Phys. Commun. 232 (2018) 124–138. https://doi.org/10.1016/j.cpc.2018.05.013
[6] J.D. Kelly, J.D. Hedengren, A steady-state detection (SSD) algorithm to detect non-stationary drifts in processes, J. Process Contr. 23 (3) (2013) 326–331. https://doi.org/10.1016/j.jprocont.2012.12.001
[7] L. Traini, V. Cortellessa, D. Di Pompeo, M. Tucci, Towards effective assessment of steady state performance in java software: are we there yet?, Empir. Softw. Eng. 28 (1) (2023) 13. https://doi.org/10.1007/S10664-022-10247-X
[8] M. Boesler, N. Weber, Steady state detection for computational fluid dynamics, arXiv:1709.03158 (2017).
[9] R.M. Bethea, R.R. Rhinehart, Applied Engineering Statistics, Routledge, 2019.
[10] M. Beseda, Kernel-based approach for precise steady-state detection in performance time series, Zenodo, v1.1.0, (2025). https://doi.org/10.5281/zenodo.16747350
[11] A. Georges, D. Buytaert, L. Eeckhout, Statistically rigorous java performance evaluation, in: Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications, OOPSLA '07, Association for Computing Machinery, New York, NY, USA, 2007, pp. 57–76. https://doi.org/10.1145/1297027.1297033
[12] T. Kalibera, R. Jones, Rigorous benchmarking in reasonable time, in: Proceedings of the 2013 International Symposium on Memory Management, ISMM '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 63–74. https://doi.org/10.1145/2491894.2464160
[13] E. Barrett, C.F. Bolz-Tereick, R. Killick, S. Mount, L. Tratt, Virtual machine warmup blows hot and cold, Proc. ACM Program. Lang. 1 (OOPSLA) (2017). https://doi.org/10.1145/3133876
[14] I.A. Eckley, P. Fearnhead, R. Killick, Analysis of Changepoint Models, Cambridge University Press, 2011, p. 205–224. https://doi.org/10.1017/CBO9780511984679.011
[15] R. Killick, P. Fearnhead, I.A. Eckley, Optimal detection of changepoints with a linear computational cost, J. Am. Stat. Assoc. 107 (500) (2012) 1590–1598. https://doi.org/10.1080/01621459.2012.737745
[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[17] W. Tang, G. Long, L. Liu, T. Zhou, M. Blumenstein, J. Jiang, Omni-scale CNNs: a simple and effective kernel size configuration for time series classification, in: International Conference on Learning Representations, 2022. https://openreview.net/forum?id=PDYs7Z2XFGv. https://doi.org/10.48550/arXiv.2002.10061
[18] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, Time Series Analysis: Forecasting and Control, Wiley, 5th edition, 2015.
[19] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, S. Tarantola, Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index, Comput. Phys. Commun. 181 (2) (2010) 259–270. https://doi.org/10.1016/j.cpc.2009.09.018
[20] R.R. Yager, On weighted median aggregation, Int. J. Uncertainty Fuzziness Knowledge Based Syst. 2 (01) (1994) 101–113. https://doi.org/10.1142/S0218488594000092
[21] R.R. Yager, Fusion of ordinal information using weighted median aggregation, Int. J. Approximate Reasoning 18 (1–2) (1998) 35–52. https://doi.org/10.1016/S0888-613X(97)10003-2
[22] K. Nehring, M. Pivato, The median rule in judgement aggregation, Econ. Theory 73 (4) (2022) 1051–1100. https://doi.org/10.1007/s00199-021-01348-7
[23] K. Khan, S.U. Rehman, K. Aziz, S. Fong, S. Sarasvady, DBSCAN: past, present and future, in: The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014), IEEE, 2014, pp. 232–238. https://doi.org/10.1109/ICADIWT.2014.6814687
[24] H.A. David, J.L. Gunnink, The paired *t* test under artificial pairing, Am. Stat. 51 (1) (1997) 9–12. https://doi.org/10.1080/00031305.1997.10473578
[25] W.J. Conover, Practical Nonparametric Statistics, 350, john wiley & sons, 1999.
[26] S.S. Shapiro, M.B. Wilk, An analysis of variance test for normality (complete samples), Biometrika 52 (3–4) (1965) 591–611. https://doi.org/10.1093/biomet/52.3-4.591
[27] M.G. Bulmer, Principles of Statistics, Courier Corporation, 2012.
[28] R. Mises, Wahrscheinlichkeitsrechnung und ihre Anwendung in der Statistik und theoretischen Physik, Mary S. Rosenberg, 1931.
[29] N.V. Smirnov, Table for estimating the goodness of fit of empirical distributions, Ann. Math. Stat. 19 (2) (1948) 279–281.
[30] T.W. Anderson, D.A. Darling, Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes, Ann. Math. Stat. 23 (2) (1952) 193–212. http://www.jstor.org/stable/2236446.
[31] H. Levene, Robust tests for equality of variances, Contrib. Probab. Stat., (1960) Stanford University Press, 278–292.
[32] M.B. Brown, A.B. Forsythe, Robust tests for the equality of variances, J. Am. Stat. Assoc. 69 (346) (1974) 364–367. https://doi.org/10.1080/01621459.1974.10482955
[33] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: a practical and powerful approach to multiple testing, J. R. Stat. Soc. Series B (Methodological) 57 (1) (1995) 289–300. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x
[34] A. Blázquez-García, A. Conde, U. Mori, J.A. Lozano, A review on outlier/anomaly detection in time series data, ACM Comput. Surv. 54 (3) (2022) 56:1–56:33. https://doi.org/10.1145/3444690
[35] J. Bai, P. Perron, Computation and analysis of multiple structural change models, J. Appl. Econom. 18 (1) (2003) 1–22. https://doi.org/10.1002/jae.659
[36] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. (CSUR) 46 (4) (2014) 44:1–44:37. https://doi.org/10.1145/2523813
[37] R.J. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, 2nd edition, 2018. https://otexts.com/fpp3/.

Martin Beseda is a Postdoctoral Researcher at the University of L'Aquila (UNIVAQ), Italy, where he focuses on detecting steady states in relation to algorithm runtime and advancing the use of quantum computers in computer science, chemistry, and physics. He earned PhD in Computational Sciences and Plasma Engineering from the Technical University of Ostrava and Université Toulouse III - Paul Sabatier in 2022. Before UNIVAQ, he worked at LCPQ in Toulouse, ICGM in Montpellier, and IT4Innovations in Ostrava. His research has been published in prestigious journals, including Physical Review A, Plasma Sources Science and Technology, and Computational and Theoretical Chemistry.

Vittorio Cortellessa is Full Professor at Department of Information Engineering, Computer Science, and Mathematics of University of L'Aquila (Italy). His research interests are in software performance and reliability, where he has experimented methodologies that span from model-driven engineering through multi-objective optimization to AI-based approaches. He has published about 150 papers on international conferences and journals in his areas of interest. He has received two 10-years most influential paper awards and five best paper awards. He has served in two editorial boards of journals, and he serves as chair and program committee member of leading conferences in his area of interest.

Daniele Di Pompeo is an Assistant Professor in the Department of Information Engineering, Computer Science, and Mathematics at the University of L'Aquila in Italy. He earned his PhD in Information and Communication Technology (ICT) in 2019. His research focuses on model-based performance analysis, software refactoring, and search-based software engineering. He has published numerous articles in top-tier international journals and conferences related to his research.

Luca Traini received his PhD in Computer Science from the University of L'Aquila, Italy, in 2021, where he is currently an Assistant Professor. His research interests include performance engineering, software engineering and AI, and empirical software engineering. He has published papers in top-tier venues such as the IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE Transactions on Software Engineering (TSE), ACM Transactions on Software Engineering and Methodology (TOSEM), and Empirical Software Engineering (EMSE). In 2024, his work received the Industrial Paper Award at the IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER).

Michele Tucci is an Assistant Professor in the Department of Information Engineering, Computer Science and Mathematics at the University of L'Aquila, Italy. He earned his PhD in Computer Science from the same university in 2021, under the supervision of Romina Eramo and Vittorio Cortellessa. From 2021 to 2023, he was a postdoctoral researcher in the Department of Distributed and Dependable Systems at Charles University, Prague. In 2023, he received the SPEC Impact Award for technical leadership. His research focuses on performance regression testing, software refactoring, and the optimization of software architectures w.r.t. quality attributes such as performance and reliability.