An In-depth Analysis of CUSUM Algorithm for the Detection of Mean and Variability Deviation in Time Series

Rayane El Sibai[1(✉)], Yousra Chabchoub[1], Raja Chiky[1], Jacques Demerjian[2], and Kablan Barbar[2]

[1] LISITE Laboratory, ISEP, 92130 Issy Les Moulineaux, France
rayane.el_sibai@etu.upmc.fr, {yousra.chabchoub,raja.chiky}@isep.fr
[2] LARIFA-EDST Laboratory, Lebanese University, Fanar, Lebanon
{jacques.demerjian,kbarbar}@ul.edu.lb

Abstract. Assessing and enhancing data quality in sensors networks is an important challenge. In fact, data recorded and sent by the sensors are often dirty, they contain noisy, erroneous and missing values. This can be due to many reasons such as sensor malfunction, uncalibrated sensor and low battery of the sensor or caused by external factors such as climatic conditions or interference. We focus in this paper on change detection in the time series data issued from sensors. We particularly address slow and gradual changes as they illustrate sensor calibration drift. For this purpose, we provide in this paper an in-depth analysis and improvement of the well known Cumulative Sum (CUSUM) control chart algorithm, as it is well adapted to small shifts detection. First, we discuss the choice of the different parameters of CUSUM in order to optimize its results based on the trade-off between the false positives and the Average Run Length (ARL_δ) needed by the algorithm to detect a process mean shift of δ. A study of the variability of the Run Length (RL_δ) is provided using simulation. Then, we introduce two improvements to the CUSUM algorithm: we propose an efficient method to estimate the starting point and the end point of the mean shift. Moreover, we adapt CUSUM to detect not only process mean deviation but also process variability deviation. All these improvements are validated by simulation and against real data stream issued from water flow-meters.

Keywords: CUSUM algorithm · Change point detection
Process variability · Sensors networks · Data quality

1 Introduction

Data streams have significantly increased with the development of the Internet of Things (IoT) and the emergence of connected objects. Large and heterogeneous collections of data are continuously generated by these objects at a high rate. They are issued from the activity of different organizations belonging to

M. R. Luaces and F. Karimipour (Eds.): W2GIS 2018, LNCS 10819, pp. 25–40, 2018.
https://doi.org/10.1007/978-3-319-90053-7_4

various domains such as healthcare, financial services, social networks, logistics and transport, and public administration.

Detecting online a change in the data stream is an important issue as it can have several interpretations depending on the application domain. In [1], the authors showed that an abrupt increase in the number of destination ports in IP traffic is an efficient criterion to detect port scan attacks. In [2], an analysis of data streams issued from a multi-temporal satellite is provided. It aims to detect land use and land cover changes. Such changes can be explained by human activities, natural conditions and development activities.

Several change detection methods have been designed by the statistics research community. According to [3], these methods can be classified into two categories. First, we distinguish algorithms designed to detect changes in the scalar parameters of an independent sequence. In this category, one can cite Shewhart control charts, geometric moving average control chart, CUSUM algorithm and Bayes-type algorithms. The second category concerns algorithms that are adapted to more complex changes such as non-additive changes in multidimensional signals. Among these methods, one can mention AR/ARMA models and the likelihood ratio. Several criteria can be considered to compare all these methods. The comparison can be based on the tolerance to false positives (false alarms), the response time of the algorithm (Run Length to detect the change), or the adaptation to progressive (or small) changes. The choice of the suitable change detection method depends on the application field and the specificity of the targeted error or change.

Statistical Process Control (SPC) is a set of techniques and tools for monitoring the quality of a process. One of the main used tools are the control chart algorithms. The statistical control chart concept was first introduced by *Walter A. Shewhart* from the Bell Telephone Laboratories in 1924. Control chart algorithms are particularly used to monitor the process stability over time by detecting a change in its parameters. The most known control chart algorithms are Shewhart, exponential/geometric moving average and CUSUM control charts. The type of the control chart depends on the number of process characteristics to be monitored. The univariate control chart is a chart of one quality characteristic, while the multivariate control chart represents more than one quality characteristic.

Control charts distinguish the habitual from the non-natural variation of the process. They signal an alarm when the process presents a suspicious deviation from a standard behavior. This deviation is defined based on two given thresholds called control limits. In general, the chart contains three elements: the plotted data corresponding to the process itself, the control limits, and the process average. By comparing the plotted data to these control limits, a decision about the stability of the process is inferred. The process is said in-control when it is stable and out-of-control otherwise. As long as the process remains in-control, the data points fall within the control limits. If a data point falls outside the control limits, we consider that the process is out-of-control. An investigation is thus needed to find and eliminate the cause(s) of the occurred change.

One of the most commonly used control chart algorithms to control the process average is the Shewhart. It evaluates the state of the process based only on the information concerning the last observation of the process while ignoring the past observations. It is efficient for the detection of large shifts, as it has a shorter response time than CUSUM in this case. However, it is insensitive to small changes. CUSUM chart overcomes this problem by using the current observed data and the historical observed data. It cumulates the impact of small deviations over time what enables it to detect small shifts in the process mean. Another control chart designed to detect a shift in the process average is the Exponentially Weighted Moving Average (EWMA) [4]. It is based on a weighted average that is updated for each received item of the stream. This moving average takes into account the current item and all the observed data. More importance (higher weight) is assigned to recent data.

The main performance measure of a control chart is the Average Run Length (ARL). Run Length (RL) is the number of observations required by the control chart algorithm to signal an alarm. When the process is in-control, the RL refers to false positives rate, and in the case of change, it qualifies the response time (reactivity) of the algorithm. RL will be discussed in details in the next section.

In this paper, we focus on the cumulative sum CUSUM algorithm, a well known control chart algorithm, proposed by Page in 1954 [5] and used to detect a shift in the process parameters. It aims to monitor the variation of the average of a process, and has the ability to detect small shifts (less than 1.5σ) from the expected average (see [6] for more details). The cumulative sum CUSUM control chart was initially proposed by Page in 1954 [5], and has been addressed in several research studies, such as [7–11].

Since the response time of CUSUM to detect a change depends on the parameters chosen beforehand, we study the impact of these input parameters on the speed of the detection. After that, we propose two improvements in order to enhance the estimation of the start time and end time of each detected error. Finally, we adapt CUSUM to detect not only process mean deviation but also process variability deviation. We validate our approach by simulation and on real data concerning water consumption with some stuck-at errors.

The explored data is issued from a flow-meter delivering water to a particular sector (geographical area) in a big French city. The flow-meters measure the flow delivered to this sector. The flow measurements for a given flow-meter are very variable. Indeed, they depend on the other flow-meters supplying the associated sector. For example, a given source associated with a particular flow-meter may suddenly stop supplying water to a sector. The latter will be delivered by its other associated peripheral sources.

Actually, the big French city is divided into several sectors, where many sensors have been deployed by the water operators on the periphery of each sector. The sensors periodically measure and send to the central monitoring system many water-related observations such as the flow, pressure, and chlore. Each of these deployed sensors has two categories of attributes: spatial attributes and non-spatial attributes. The spatial attributes depict the geographical location of

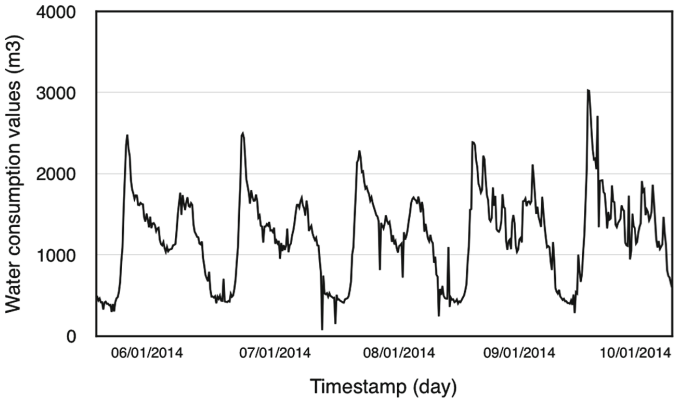

Fig. 1. Volume of the water consumed by the sector, over time

the sensor: latitude and longitude, while the non-spatial attributes include the name, ID, and the record observations of the sensor.

The data recorded by the sensors are structured data streams and have both spatial and temporal characteristics. In addition to the geographical position of the sensor, each record observation is composed of two fields: the timestamp designating the recording date of the measure, and the value of the measure. These data are regularly generated by the sensors with a frequency of one observation each 15 min.

Figure 1 shows the volume of the consumed water of a specific sector during five working days in January 2014. The volume of the water consumed by a given sector can be simply inferred in real-time as an algebraic sum of the flows delivered by its associated flow-meters, in m^3. One can notice a periodicity in the water consumption, related to the human activity.

The paper is organized as follows. Section 2 discusses the CUSUM algorithm and the choice of its parameters, and illustrates the proposed improvements. Section 3 presents the experiments performed to validate the efficiency of the proposed improvements. The paper ends with a conclusion.

2 CUSUM Algorithm

2.1 Algorithm Description

Applied on a data stream, CUSUM takes into account all the past values of the stream by calculating the cumulative sum of the deviations from the target value which is defined as the mean of the observables in the training window, μ_0. It is implicitly assumed that the observed process $(S_t)_{t\geq 0}$ is in-control (stable) during the training window. The cumulative sum control chart C_t, initially set

to 0, ($C_0 = 0$), is calculated as follows:

$$C_t = \sum_{j=1}^{t} (s_j - \mu_0); t \geq 1$$

In order to quantify and detect small variations, CUSUM defines two statistics C_t^+ and C_t^-. C_t^+ accumulates for relatively high values of the observed process, the distance to $(\mu_0 + K)$; K being a given threshold that will be discussed later. For small values of the observed process, the cumulative distance to $(\mu_0 - K)$ is handled by C_t^-.

$$C_t^+ = max[0, s_t - \mu_0 - K + C_{t-1}^+]$$
$$C_t^- = max[0, \mu_0 - s_t - K + C_{t-1}^-]$$

The threshold K is also called the allowance value. It depends on the mean shift that we want to detect. If either C_t^+ or C_t^- exceeds the decision threshold H, the process is considered as out-of-control and an alarm will be signaled. The process is declared as in-control when the cumulative sum is again under the threshold H. K and H are often related to the standard deviation σ_0 calculated in the training window:

$$K = k\sigma_0; \; H = h\sigma_0$$

After the detection of an error, the cumulative sums C_t^+ and C_t^- are reinitialized to 0.

In an improved version of CUSUM called FirCUSUM (for Fast Initial Response) [11], a headstart is introduced to improve the response time of the algorithm. When the process is out-of-control at the start-up, or when it is restarted after an adjustment, the standard CUSUM may be slow in detecting a shift in the process mean that is present immediately after the start-up of the adjustment. To overcome this problem, the headstart consists of setting the starting values C_t^+ and C_t^- equal to some nonzero value, typically $H/2$.

2.2 Choice of the Parameters

The performance of the CUSUM algorithm is closely dependent on the choice of the two key parameters h and k. Two objectives must be achieved when setting these parameters. On the one hand, one must minimize the false positives. In other words, when the process is in-control, ideally, the CUSUM algorithm should not detect any change. On the other hand, any mean shift has to be detected as soon as possible. There is clearly a trade-off between these two objectives. According to [6], it is recommended to take $k = 0.5$ and $h = 4$ *or* 5. A more complete theoretical study of the performance of CUSUM is provided by Siegmund in [12]. It is based on an approximation of the Average Run Length (ARL) properties.

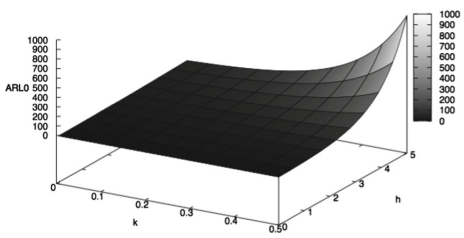

Fig. 2. $ARL_0(k, h)$: impact of k and h on ARL_0

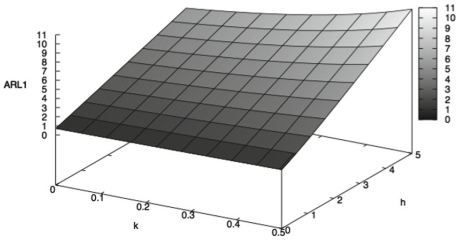

Fig. 3. $ARL_1(k, h)$: impact of k and h on ARL_1

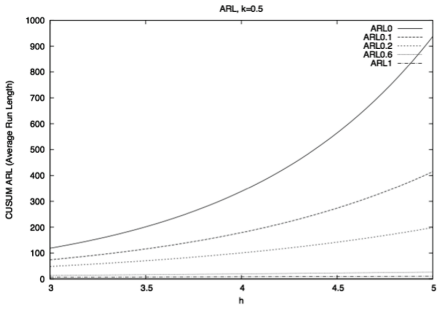

Fig. 4. Impact of the control limit h on ARL_δ

ARL_δ is defined as the expected number of items required by CUSUM to detect a deviation when the process has a mean deviation of δ. The approximation given by [12] is simple compared to other approaches based on approximating transitions from the in-control to the out-of-control state with a Markov chain (see [13]). We choose to focus on positive shifts (the problem is completely analog for negative shifts) so we consider a one-sided CUSUM where only C_i^+ is handled. In this case, Siegmund's approximation of ARL is:

$$ARL = \frac{\exp(-2(\delta - k)h') + 2(\delta - k)h' - 1}{2(\delta - k)^2}$$

where δ is the mean process shift, in the units of σ_0, and $h' = h + 1.166$. In this equation, the observed process is assumed to be normally distributed.

We first plotted in Fig. 2 the variation of ARL_0 based on this equation. Recall that ARL_0 should be high to minimize false positives. According to this Figure, h must be taken at least equal to 3 to have an ARL_0 higher than 100, when k is close to 0.5.

The second step is to minimize ARL_δ to detect the deviation as soon as possible and to achieve a good reactivity. According to Fig. 3, the ARL decreases when h decreases, that is why h should be small to have a better reactivity in case of a mean shift. For $h \in [3, 5]$, ARL_1 is between 5 and 11 which corresponds

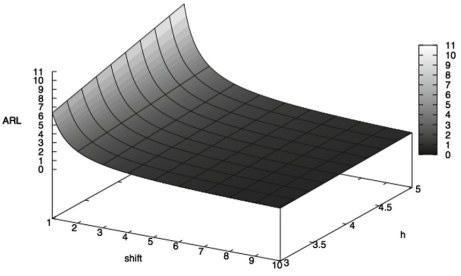

Fig. 5. Impact of the shift δ on ARL, $h \in [3, 5]$

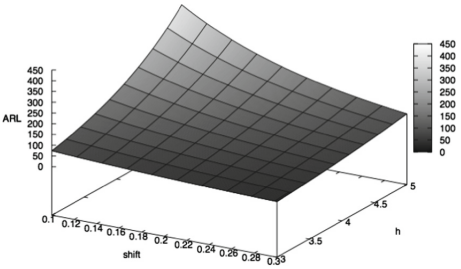

Fig. 6. Impact of small shifts δ on ARL, $h \in [3, 5]$

to a reasonable response time or reactivity. This result is confirmed by Fig. 5 which shows that CUSUM has a small ARL_δ for large shifts ($\delta \geq 1$).

In Figs. 4 and 6, we focused on very small shifts detection ($\delta \in [0, 1]$). When the mean deviation δ is small, ARL_δ becomes high, close to ARL_0. One can conclude that the shift δ has to be at least equal to 0.5 to guarantee an order of magnitude of difference between ARL_0 and ARL_δ ($ARL_0 \sim 100$ and $ARL_\delta \sim 10$).

2.3 Variability of the Run Length

An important criterion used to evaluate the reactivity of a control chart algorithm is the Average Run Length (ARL). We recall that it represents the expected number of observations needed before an out-of-control alarm is detected. ARL has two interpretations. ARL_0 is defined as the average number of in-control data needed by the control chart algorithm to signal a false alarm. ARL_δ is the average number of out-of-control data required to detect the error after a process mean changed. ARL_0 has to be as large as possible, and ARL_δ has to be small in order to detect the shift quickly. The run length is a random variable. To the best of our knowledge, only its average was theoretically studied. No theoretical results about the variability of Run Length are provided in the literature. In this paper, We address this problem using simulations.

We report in Table 1 the average and the standard deviation of the Run Length, respectively ARL and $SDRL$, for different values of shift δ. In order to simulate ARL_0, a sample of size 1000 is generated under an in-control situation. CUSUM chart is then applied to the samples until an out-of-control signal is triggered. The number of observations when the signal is triggered is the in-control Run Length RL_0. We repeat this simulation 150 times so we can get the average value: ARL_0. The considered process follows the standard normal distribution. To simulate the ARL_δ, we performed the same experiments with an injected shift of δ in the process mean. We can notice that the average and the standard deviation of the Run Length decrease with the increase of the shift δ. Moreover, the false positives detected by CUSUM closely depend on the execution, as they are calculated using RL_0 which has a high standard deviation

Table 1. Simulated Run Lengths (RL), for different shift sizes

	Shift δ					
	0	0.25	0.5	1	2	3
Simulated ARL	328	73.59	25.82	8.02	3.38	2.74
SDRL	220	8.38	1.92	1.77	1.41	1.31

compared to its average ARL_0. Therefore the Run Length to obtain a false positive can be significantly lower than ARL_0. One can conclude that for a single execution, the CUSUM can lead quickly to a false positive.

2.4 Enhancing the Reactivity of CUSUM Algorithm

Recall that to detect positive mean shifts, CUSUM is based on the following cumulative statistic:

$$C_t^+ = max[0, s_t - \mu_0 - K + C_{t-1}^+]$$

In the standard version of CUSUM, the deviation from the expected average is declared after performing enough iteration to deeply impact C_t. However, it is not possible to know the exact start time of the small shift. In fact, the change detection occurs after real change start time and no estimation of this latter is provided in the literature. Moreover, the observed process is considered as in-control (end of the deviation) when C_t is less than the detection threshold H. As C_t is a cumulative sum, it sometimes takes a long time (many iterations) to achieve this condition. Thus, the end of deviation is declared a long time after the real return to the standard behavior.

To obtain a complete CUSUM algorithm, we propose the following improvements:

- Add an estimation of the start time of the change.
- Improve the precision of the end time of the change.

Error Start Time. The error real start time estimation for CUSUM algorithm was not addressed before. The change is simply declared when it is detected. Let us take $K = \sigma_0/2$, and s_t a process normally distributed: $S \sim \mathcal{N}(\mu_0, \sigma_0^2)$.

When the process s_t is in-control, $(s_t - \mu_0 - K)$ has the same probability of being positive or negative for symmetry reasons. When s_t has a mean positive shift (of σ_0 as an example), $(s_t - \mu_0 - K)$ becomes very likely to be positive. Therefore, C_t is very likely to be increasing $(C_t > C_{t-1})$. This is the key idea behind our improvement. When the process is out-of-control, the value of C_t is very likely to increase as long as the deviations persist.

When an error is detected $(C_t > H)$, the start time of this error can be estimated by the moment where C_t became strictly increasing. This moment can be inferred even if it is former to the detection of the error. For this purpose, we

introduce a counter N that we update each time we calculate C_t. If an error is declared at time t, its start time is estimated as: error start time $= t - N + 1$.

Initially, N is set to 0, then, it is updated as follows:

- $N \leftarrow N + 1$ if the value of C_t increases.
- $N \leftarrow N - 1$ if the value of C_t decreases.
- N is reset to 0 if the end of the error is detected.

In the case of an in-control process, N has a random distribution with a null mean. The counter N is used to estimate the size of each error, in other words, the process out-of-control duration.

Error End Time. The second improvement of CUSUM is related to the end of the error. In the standard version of CUSUM, the end of the error is declared when C_t becomes lower than the threshold H. Being a cumulated sum, C_t needs many steps (or iterations) to attain its normal values ($<H$) after the end of the error. To improve the reactivity of the CUSUM algorithm, we introduce a counter Z to be able to detect the end of the error quickly.

The key idea of this improvement is that when C_t becomes constant or decreasing, the current deviation is very likely to be stopped. The condition $C_t^- \leq C_{t-1}^-$ is always achieved before $C_t^- < H$ as in case of error $C_{t-1}^- > H$.

The counter Z is updated each time we calculate C_t. It depicts the number of successive decreases of C_t.

Initially, Z is set to 0, then, it is updated as follows:

- $Z \leftarrow Z + 1$ if C_t decreases.
- $Z \leftarrow 0$ otherwise.

The end of the error is declared when Z exceeds a given threshold Z_0. Just like σ_0 and μ_0, this latter is inferred from the training window. It is the average number of successive decreases in the training window (when the process is in-control).

The detailed pseudo code of the anomaly detection using the improved CUSUM algorithm is presented in Algorithm 1.

3 Experiments and Results

3.1 Efficiency Metrics

In this section, we evaluate the performance of CUSUM algorithm for the change detection. In information retrieval domain, the performance metrics used to evaluate the performance of a change detector are precision, recall, and specificity.

These metrics are based on the True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). The errors committed by CUSUM can either be false positives (the process is considered as out-of-control while it is not true) or false negatives (no alarm is signaled whereas the process is out-of-control). *Precision* metric is the proportion of true alarms compared to

all the alarms risen by CUSUM. *Recall* metric depicts the true positive rate: the probability that CUSUM algorithm identifies a truly erroneous point. *Specificity* metric represents the true negative rate: the proportion of points considered truly as not erroneous by CUSUM compared to the total number of not erroneous points present in the dataset.

$$Precision = \frac{TP}{TP + FP};\ Recall = \frac{TP}{TP + FN};$$
$$Specificity = \frac{TN}{TN + FP};$$

Algorithm 1. Improved CUSUM anomaly detection algorithm

Input : Stream of items S $\{s_0, s_1, ..., s_n\}$, $K = \sigma_0/2$, $H = 4\sigma_0$
Output: errorAlarm(), errorStartTime, errorEndTime

```
1  C_0 ← 0; N ← 0; Z ← 0;
2  errorInProgress ← false;
3  foreach incoming item s_t, t > 0 do
4      C_t = max[0, s_t − μ_0 − K + C_{t−1}]
5      if (C_t > C_{t−1}) then
6          N++;
7          Z = 0;
8          if (C_t > H) then
9              errorAlarm();
10             errorStartTime = t − N + 1;
11             errorInProgress = true;
12         end
13     end
14     if (C_t < C_{t−1}) then
15         N−−;
16         Z++;
17         if (errorInProgress == true) then
18             if (Z > Z_0) then
19                 errorEndTime = t − 1;
20                 errorInProgress = false;
21                 C_t = 0;
22                 N = 0;
23             end
24         end
25     end
26 end
```

3.2 Detecting Mean Change

The objective of this section is to validate our proposed improvements of CUSUM algorithm to detect a negative shift of the mean among normally distributed simulated data. Let us take a process $(S_t)_{t\geq 0}$ that follows the standard normal distribution: $S \sim \mathcal{N}(0, 1)$. We first considered 1000 observations of S, then we injected at random moments 10 deviations (also called errors). The purpose of the experiments is to apply CUSUM to detect these deviations and to use our improvements to estimate the start time and the end time of each detected deviation. The errors have a random length taken in $[1, 50]$. The cumulative errors length equals 239 points or observations. As the targeted change is a mean shift, for each error, we replaced the original points by new observations issued from a shifted process $S' \sim \mathcal{N}(-1, 1)$. Notice that the variance of the process remains unchanged.

As only negative shift is considered in this section, we only focus on the cumulative parameter C_t^-, that we simply denote by C in the following parts.

$$C_t^- = max[0, \mu_0 - s_t - K + C_{t-1}^-]$$

k is set to $0,5$ and the threshold h is taken equal to 4, according to the recommendations given in Sect. 3.

The variation of C over time, together with the detection threshold H are plotted in Fig. 7. We can notice that C is very variable over time and presents several peaks that are mainly caused by the injected errors. CUSUM signals errors each time the value of C exceeds the control limit H. We obtained a total number of 8 alarms corresponding to the 8 detected errors. Two errors are missed because they have very small durations.

Figure 8 depicts the variation of the counter N over time. Recall that N is used to estimate the start time of the error. It is incremented by one with the increase of C and decremented C decreases. Figure 8 shows that when the process is in-control, the value of N has a null average and very small standard deviation. The presence of an error induces a notable increase of N.

The variation of the counter Z over time is shown in Fig. 9. Z counter enables to estimate the end time of the error. It counts the number of successive decreases of C. We can see that Z has small variations with a mean of 0.22 and a standard deviation of 0.57. The end of the error is declared when Z exceeds its average value Z_0 calculated in the training window. In our case, Z_0 equals to 0.25.

The evaluation of the proposed improvements is given in Table 2. The 1000 considered points of the process are first classified into actual error and actual not error. Then we add a second classification according to the detection results of the improved CUSUM. The same experiments are performed with the standard version of CUSUM and FirCUSUM. Recall that FirCUSUM is an improved version of CUSUM. With the classic version of CUSUM algorithm, the values C_t^+ and C_t^- are reset to zero after the detection of a change. The objective of FirCUSUM is to improve the performance of CUSUM by setting the values C_t^+ and C_t^- equal to some nonzero value, typically $H/2$.

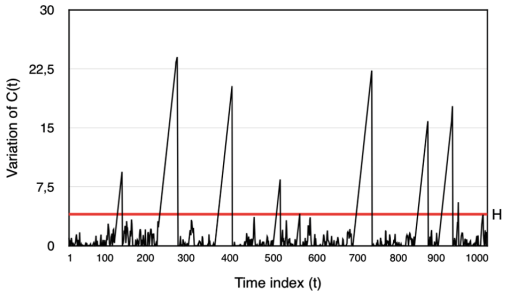

Fig. 7. Variation of C_t over time

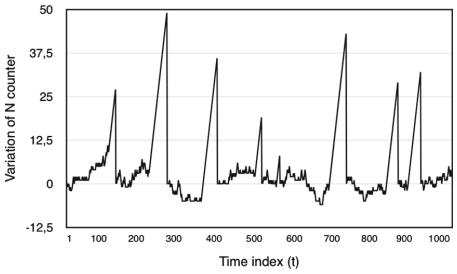

Fig. 8. Variation of N over time

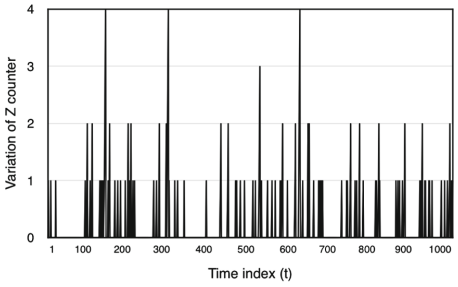

Fig. 9. Variation of Z counter over time

Table 2. Obtained results for mean change detection

CUSUM version	Actual	Detected as	
		Error	Not error
Standard	Error	189	50
	Not error	203	558
FirCUSUM	Error	174	65
	Not error	196	565
Improved	Error	**221**	**18**
	Not error	**23**	**738**

The obtained results are presented in Table 2. One can notice that our proposed algorithm outperforms both standard and FirCUSUM. It significantly decreased both false negatives and false positives.

The three efficiency metrics (precision, recall, and specificity) of the standard CUSUM, FirCUSUM and the improved version of CUSUM are given in Table 3. All these metrics are enhanced using our improved version of CUSUM. The proposed improvements give very good results as the three efficiency metrics are above 0.9.

Table 3. Performance metrics of CUSUM

CUSUM version	Precision	Recall	Specificity
Standard	0.48	0.79	0.73
FirCUSUM	0.47	0.72	0.74
Improved	**0.90**	**0.92**	**0.96**

3.3 Application to Stuck-at Error: Detecting Variation Change

In this section, we apply the improved CUSUM algorithm on a real data stream issued from water flow-meters, to detect the so-called stuck-at errors.

During the data analysis process, the conclusions and decisions are based on the data. If the data are dirty, this will leads to defective and faulty results. Improving the data quality is thus inevitable to obtain reliable results. One of the data quality measures is the *accuracy*. It represents the difference between the observation's value and the true value which the sensor aims to represent. Due to the instrumental, physical and human limitations, malfunction and miscalibration of the sensor, the observations values of the sensor can deviate significantly from the true ones. These deviated values are called faults [14].

The stuck-at error, also called CONSTANT in [15,16], is a type of sensor errors. A stuck-at x error occurs when the sensor is stuck on an incorrect value x. The low battery of the sensor, a dead sensor, or the malfunction of the sensor, may cause this error. During such a situation, a set of successive observations will have the same value $x \pm \epsilon$. The error may last a long time, and the sensor may or not return to its normal behavior. [16] showed that CONSTANT errors are present in the sensors data of the INTEL Lab and in NAMOS data set. This kind of error concerns about 20% of their data. The variance is the characteristic to be modeled in order to detect this type of error, during which, the variance of the data drops significantly.

We use the improved CUSUM to detect the stuck-at errors. As the change concerns the variance of the observed process, we choose to apply CUSUM on the variations v_t of the observed values: $v_t = |s_t - s_{t-1}|$. $(V_t)_{t\geq 0}$ is a positive time series with an average of μ_0 and a standard deviation σ_0 during the training window. As the stuck-at error engenders a decrease of μ_0, we only focus here on the cumulative statistic C_t^-, for a one-sided CUSUM.

The dataset duration considered in this section is of 10 days in January 2014, with a total number of 960 observations. To inject a stuck-at error in the time series, we chose random instant t and we replaced s_t by a random variable uniformly distributed in $[s_t - \mu_0 + \sigma_0, s_t + \mu_0 - \sigma_0]$, for a random number of successive values beginning from the instant t. Hence the mean of the process $v_t = |s_t - s_{t-1}|$ drops from μ_0 to $\mu_0 - \sigma_0$. The shift that we want to detect is about $-\sigma_0$. We repeated this mechanism 10 times to inject 10 errors. The errors have a random length taken in $[1, 50]$. We obtained a total number of 263 injected erroneous points.

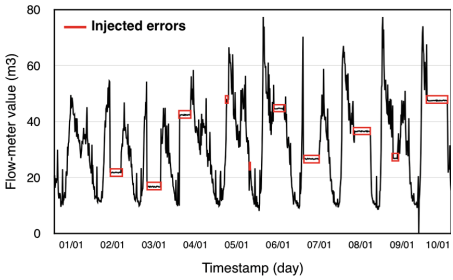

Fig. 10. Injection of variation changes

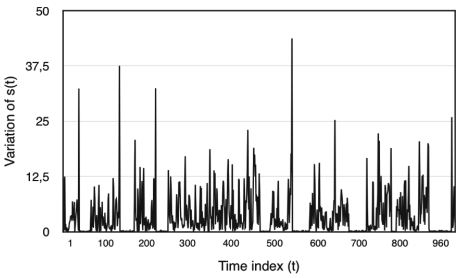

Fig. 11. Variation v_t of s_t after the injection of errors

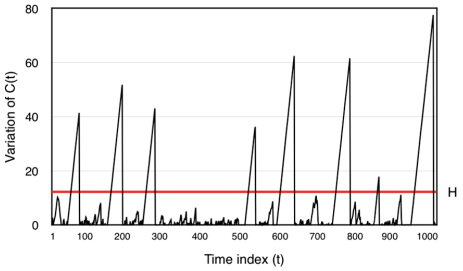

Fig. 12. Variation of C_t over time

Figure 10 shows the considered dataset s_t with the injected errors. The variations of s_t denoted as $v_t = |s_t - s_{t-1}|$ are plotted in Fig. 11. One can easily notice a periodicity in the water consumption with a difference between the working and not working days. Moreover, during the injected errors, the variations of s_t denoted as $v_t = |s_t - s_{t-1}|$ drop significantly. Recall that we performed CUSUM on v_t.

Figure 12 displays the variation of C_t over time. 8 alarms were signaled by CUSUM, when C_t exceeds the threshold H. They correspond to real injected errors. CUSUM missed two errors as they have very small durations (of only 1 and 4), compared to ARL_1. In fact, according to Sect. 3.2, ARL_1 equals 8.38 (when $k = 0.5$, $\delta = \sigma$ and $h = 4$). It means that we need in average 8 observations to could detect this change.

We recap in Tables 4 and 5 the obtained results of the variation change detection, after applying the improved and the standard version of CUSUM algorithm. Just like in the previous subsection (using simulation), we checked, based on these values that the proposed improvements enhance the three efficiency metrics: the precision, the recall and the specificity of the CUSUM algorithm (see Table 5). Moreover, these results show that the CUSUM algorithm performs good results even if the considered data set does not verify the normal distribution.

Table 4. Obtained results for stuck-at errors detection

CUSUM version	Actual	Detected as	
		Stuck-at error	Not stuck-at error
Standard	Stuck-at error	134	129
	Not stuck-at error	121	576
Improved	Stuck-at error	**255**	**8**
	Not stuck-at error	**68**	**629**

Table 5. Performance metrics of CUSUM

CUSUM version	Precision	Recall	Specificity
Standard	0.52	0.50	0.82
Improved	**0.78**	**0.97**	**0.90**

4 Conclusion

We presented in this paper an in-depth study of CUSUM control chart algorithm. We first analyzed the average and the variations of the Run Length, which qualifies both false positives and the response time of the algorithm in case of a change detection. Then, we introduced an improvement of the precision of the estimation of the start time and the end time of each detected change. The designed algorithm performs good results against both synthetic and real datasets. We also adapted the CUSUM to deal with the stuck-at error where the variability of the process is addressed.

References

1. Chabchoub, Y., Chiky, R., Dogan, B.: How can sliding HyperLogLog and EWMA detect port scan attacks in IP traffic? EURASIP J. Inf. Secur. **2014**(1), 5 (2014)
2. Manonmani, R., Mary Divya Suganya, G.: Remote sensing and GIS application in change detection study in urban zone using multi temporal satellite. Int. J. Geomat. Geosci. **1**(1), 60 (2010)
3. Basseville, M., Nikiforov, I.V., et al.: Detection of Abrupt Changes: Theory and Application, vol. 104. Prentice Hall, Englewood Cliffs (1993)
4. Roberts, S.W.: Control chart tests based on geometric moving averages. Technometrics **1**(3), 239–250 (1959)
5. Page, E.S.: Continuous inspection schemes. Biometrika **41**(1/2), 100–115 (1954)
6. Montgomery, D.C.: Introduction to Statistical Quality Control. Wiley, New York (2007)
7. Ewan, W.D.: When and how to use CUSUM charts. Technometrics **5**(1), 1–22 (1963)
8. Bissell, A.F.: CUSUM techniques for quality control. Appl. Stat. **18**, 1–30 (1969)
9. Goel, A.L., Wu, S.M.: Economically optimum design of CUSUM charts. Manag. Sci. **19**(11), 1271–1282 (1973)

10. Reynolds, M.R.: Approximations to the average run length in cumulative sum control charts. Technometrics **17**(1), 65–71 (1975)
11. Lucas, J.M., Crosier, R.B.: Fast initial response for CUSUM quality-control schemes: give your CUSUM a head start. Technometrics **24**(3), 199–205 (1982)
12. Siegmund, D.: Sequential Analysis: Tests and Confidence Intervals. Springer, New York (2013). https://doi.org/10.1007/978-1-4757-1862-1
13. Brook, D., Evans, D.A.: An approach to the probability distribution of CUSUM run length. Biometrika **59**, 539–549 (1972)
14. El Sibai, R., Chabchoub, Y., Chiky, R., Demerjian, J., Barbar, K.: Assessing and improving sensors data quality in streaming context. In: Nguyen, N.T., Papadopoulos, G.A., Jędrzejowicz, P., Trawiński, B., Vossen, G. (eds.) ICCCI 2017. LNCS (LNAI), vol. 10449, pp. 590–599. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67077-5_57
15. Ramanathan, N., Schoellhammer, T., Estrin, D., Hansen, M., Harmon, T., Kohler, E., Srivastava, M.: The final frontier: embedding networked sensors in the soil. Center for Embedded Network Sensing (2006)
16. Sharma, A.B., Golubchik, L., Govindan, R.: Sensor faults: detection methods and prevalence in real-world datasets. ACM Trans. Sens. Netw. (TOSN) **6**(3), 23 (2010)