



Noticing the Slide : Calibrated Drift Detection for Deployed Agents

Appiah Prince ¹ , Vijayalakshmi Kakulapati ²

ITMO University, Saint Petersburg, Russia; princeappiah35@gmail.com

Department of Computer Science and Engineering , MVSR Engineering College, Hyderabad, Telangana, India;
vldms@yahoo.com

* Corresponding Author: Vijayalakshmi Kakulapati ; vldms@yahoo.com

Abstract: A deployed agent rarely breaks all at once. A model update, a changed tool, or a shift in what users ask for can raise its violation rate a little at a time, and nobody notices until the damage is done. We ask how quickly different change detectors notice such drift when each is set to the same false-alarm rate. We compare a CUSUM, a likelihood-ratio CUSUM, an EWMA chart, a rolling-mean threshold, a sliding-window test and an ADWIN-style adaptive window on streams of per-task violation indicators, with every detector calibrated to one false alarm per 1,000 tasks and checked on fresh data. For a step increase from 5% to 10%, the CUSUM family, the EWMA chart and the adaptive window all needed about 69 to 75 tasks (median), the rolling mean 79, and the sliding-window test 120. On bursty drift the gap widens to 4.8 times. For a slow ramp, the classical first-order delay formula was wrong by up to 9 times. We derive a better one, treating the CUSUM as a reflected random walk with slowly changing drift, and it predicts the simulated median delay within 8% over a hundredfold range of slopes. Estimating the baseline from 500 tasks raised false alarms by up to 1.39 times for detectors that use it, and benign seasonality of 100% amplitude raised them about threefold. All detectors ran in under 2 μ s per observation. All results are simulation-based.

Keywords: Drift Detection, CUSUM, EWMA, ADWIN, Agent Monitoring, False-Alarm Calibration.

1. Introduction

Software that learns, or that is built on something that learns, can get worse without any single event that looks like a failure. A provider updates a model behind an API. A tool the agent relies on changes its output format. Users start asking different questions. Each of these can nudge the agent's rate of errors or policy violations upward by a fraction of a percentage point, and for weeks the dashboard still shows a number that looks fine. Statisticians have studied exactly this problem for seventy years under the name of change detection [1, 6]. Machine-learning researchers have studied a close relative as concept drift [8, 9]. What is missing, in our experience, is a plain comparison of the standard tools on the kind of data agent operators actually have, which is a stream of mostly zeros with a few ones, under conditions that match real deployments: drift that is slow or intermittent, a baseline that has to be estimated, and traffic that varies for harmless reasons. The comparison has one trap. A detector that raises more alarms looks faster, so any fair comparison must hold the

false-alarm rate fixed. We set every detector to the same average time between false alarms and verify it on separate data before measuring anything else. This is the single most important step in the paper, and the one most often skipped.

We make four contributions.

- A calibrated comparison of six detectors on step, ramp, creep and bursty drift, with false-alarm rates verified on fresh data (Sections 3 and 6).
- A finite-sample check of the classical delay formula, which we find overstates delay at moderate false-alarm intervals, and a new delay law for ramp drift that is accurate to within 8% in our tests (Section 4).
- Measurements of how an estimated baseline and benign seasonality inflate false alarms, and of how much a seasonal correction recovers.
- Measured per-observation costs of simple scalar implementations, which are small enough to ignore.



- Everything is simulated. The streams are synthetic Bernoulli sequences with chosen drift patterns, not logs from real agents.

2. Background and Related Work

Page introduced the cumulative-sum chart in 1954 [1]. Lorden showed that it is optimal in a worst-case sense when the post-change distribution is known [2], and Moustakides gave the exact optimality proof [3]. Lai extended the information bounds to general stochastic processes and gave the asymptotic relation between the false-alarm interval and the best possible delay [4]. Roberts proposed the exponentially weighted moving average chart as a simpler alternative that responds to small shifts [5]. Basseville and Nikiforov's book collects the theory [6], and Siegmund's gives the approximations to average run length that we use below [7].

In machine learning, concept drift means that the relationship between inputs and outputs changes over time. Surveys [8, 9] organize detectors by what they monitor: errors, input distributions or model confidence. Window-based detectors compare a recent window with an older one. ADWIN [10] does this adaptively, searching over window splits and using a Hoeffding-type bound to decide whether two sub-windows differ. Its guarantees are valid, though conservative, and we show below what that conservatism costs. Our contribution is not a new detector. It is a careful comparison at matched false-alarm rate on streams that resemble agent telemetry, and an examination of where the textbook formulas stop being accurate.

3. Research Gap and Methodology

3.1. Stream Model

At each task t the agent produces an outcome $x_t \in \{0, 1\}$; where 1 means a violation or failure. Before drift, $x_t \sim \text{Bernoulli}(p_0)$; $p_0 = 0.05$ with $p_0 = 0.05$. At an unknown time t_0 the probability starts to change. We want to raise an alarm soon after t_0 without raising many before it. Figure 1 shows where the detector sits in a monitoring pipeline, and Figure 2 shows the drift patterns we test.

3.2. Detectors

CUSUM (mean form) keeps $S_t = \max\{0, S_{t-1} + x_t - p_0 - k\}$ and alarms when $S_t > H$, with reference value $k = 0.025$ (half the shift we design for). CUSUM-LLR adds the log-likelihood ratio of a design alternative $p_1 = 0.10$ and alarms above h . EWMA smooths with weight $\lambda = 0.02$ and $S_t > H$; $k = 0.025$ alarms when the smoothed value exceeds p_0 by L standard deviations. Rolling mean alarms when the mean of the last 100 outcomes exceeds a threshold τ . Window z -test compares the last 100 outcomes with the 400 before them using a two-proportion z statistic and $W_t = \max\{0, W_{t-1} + \ell(x_t)\}$; alarm if $W_t > h$. ADWIN-style compares

the most recent s outcomes ($s = 32, 64, 128, 256, 512$) with the 512 before them and alarms $(1/w) \sum_{i=0}^{w-1} x_{t-i} > \tau$, if the mean difference exceeds a Hoeffding-type cut scaled by a factor c . We call it "style" because the original also shrinks the window and uses exact bookkeeping $Z_t = \lambda x_t + (1 - \lambda)Z_{t-1}$, and we scale the cut for a reason explained next. Detectors reset after an alarm, and window detectors pause for a cooldown equal to their window. Say the following [2, 4]. The LLR-CUSUM with threshold h has $ARL_0 \geq e^h$, and $Z_t > p_0 + L\sqrt{p_0(1-p_0)\lambda/(2-\lambda)}$ its worst-case expected delay is asymptotically $h/D(p_1/p_0)$ as $h \rightarrow \infty$. Combined, any detector with $ARL_0 \geq \gamma$ has worst-case delay at least $\ell(x_t) = x_t \ln(p_1/p_0) + (1-x_t) \ln[(1-p_1)/(1-p_0)]$. We state these without proof because they are known.

$$z = (\hat{p}_1 - \hat{p}_0) / \sqrt{p(1-p)(1/w_1 + 1/w_0)}$$

$$|\hat{p}_{\text{recent}} - \hat{p}_{\text{old}}| > c \times (\text{Hoeffding-type cutoff})$$

3.3. Calibration

We set the free threshold of each detector so that the average run length to a false alarm, ARL_0 , equals 1,000 tasks. This is done by bisection on 800 simulated streams of 5,000 tasks with no drift, and we then verified it on 1,500 new streams of 6,000 tasks. Table 1 gives the settings and the verified ARL_0 . Everything is within 15% of target. One adaptation needed explaining. The unscaled AD-WIN cut is so conservative that even the loosest setting ($c = 1$ with $\delta = 0.9$) gave ARL_0 near 7,900, so it could not be brought down to 1,000. We therefore calibrate a scale factor c on the cut, and found $c = 0.35$. This means the Hoeffding-type bound is about a factor of three wider than needed for this false-alarm rate. It is a modification of the original method, and results for "ADWIN-style" should be read with that in mind. The practical question is how accurate they are at the moderate $\gamma = 1,000$ and $p_0 = 5\%$ that matter here. For a step to $p_1 = 0.10$, $D = 0.0207$ and the asymptotic bound is $\ln(1000)/D = 334$ tasks. The calibrated LLR threshold is only $h = 2.75$, whereas the bound $ARL_0 \approx e^h$ would suggest $h = \ln 1000 = 6.9$. The inequality is very loose here: $e^h = 16$ against a true ARL_0 of 1,000. A threshold of 2.75 gives a delay of about $h/D = 133$ in the asymptotic formula, and the simulated median delay is about half of that (69), because the formula is for the worst-case starting state and our onsets occur in steady state. The lesson is that the textbook expression $\ln \gamma/D$ should not be used as a delay estimate at moderate false-alarm intervals for rare events. It overstated the delay about five-fold here $ARL_0 = 1,000$ tasks.

4. Related Work

4.1. The classical relation and its finite-sample accuracy

For a Bernoulli stream, the Kullback–Leibler diver- $ARL(\mu) = 2\mu 2e^{+} \sigma^2 - 1$, (2) gence between the post-change rate p_1 and baseline p_0 is $D(p_1 \parallel p_0) = p_1 \ln(p_1/p_0) + (1 - p_1) \ln[(1 - p_1)/(1 - p_0)]$. Classical results say the following [2, 4]. The LLR-CUSUM with threshold h has $ARL_0 \geq e^h$, and its worst-case expected delay is asymptotically Worst-case expected delay $\sim h / D(p_1 \parallel p_0)$, as $h \rightarrow \infty$. Combined, any detector with $ARL_0 \geq \gamma$ has Worst-case delay $\geq (1 + o(1)) \ln(\gamma) / D(p_1 \parallel p_0)$. The practical question is how accurate they are at the moderate $\gamma = 1,000$ and $p_0 = 5\%$ that matter here. For a step to $p_1 = 0.10$, $D = 0.0207$ and the asymptotic bound is $\ln D(0.10 \parallel 0.05) \approx 0.0207$; $\ln(1000)/D \approx 334$ tasks. The

calibrated LLR threshold is only $h = 2.75$, whereas the bound $ARL_0 \geq e^h$ would suggest $h = \ln 1000 = 6.9$. The inequality is very loose here: $e^h = 16$ against a true ARL_0 of 1,000. A threshold of 2.75 gives a delay of about $h/D = 133$ in the asymptotic formula, and the simulated median delay is about half of that (69), because the formula is for the worst-case starting state and our onsets occur in steady state. The lesson is that the textbook expression $\ln \gamma/D$ should not be used as a delay estimate at moderate false-alarm intervals for rare events. It overstated the delay about five-fold here.

Table. 1 Detector settings calibrated to $ARL_0 = 1,000$, and ARL_0 verified on fresh data.

Detector	Fixed setting	Calibrated	ARL_0
CUSUM	$k = 0.025$	$H = 3.55$	1006
CUSUM-LLR	$p_1 = 0.10$	$h = 2.75$	997
EWMA	$\lambda = 0.02$	$L = 2.6$	984
Rolling mean	$w = 100$	$\tau = 0.09$	1115
Window z-test	$w_1 = 100, w_0 = 400$	$z_c = 1.83$	1022
ADWIN-style	$\delta = 0.01$, old window 512	$c = 0.354$	967

4.2. A Delay Law For Ramp Drift

Suppose the violation probability rises linearly after t_0 , so $p_s = p_0 + rs$ at s tasks after onset. Consider the mean-form CUSUM with reference k and threshold H .

Proposition 1 (First-order law). *If noise is ignored, the statistic grows when $rs > k$, and the expected accumulated drift $rs^2/2 - ks = H$ reaches H at*

$$\tau_f = [k + \sqrt{(k^2 + 2rH)}] / r$$

This is the natural first guess, and it is wrong for slow ramps. It assumes the statistic does not move until the drift beats the reference value k , but noise alone can carry it over the threshold. The statistic is a random walk reflected at zero with drift $\mu_s = rs - k$ per task and variance $\sigma^2 = p_0(1 - p_0)$ per task.

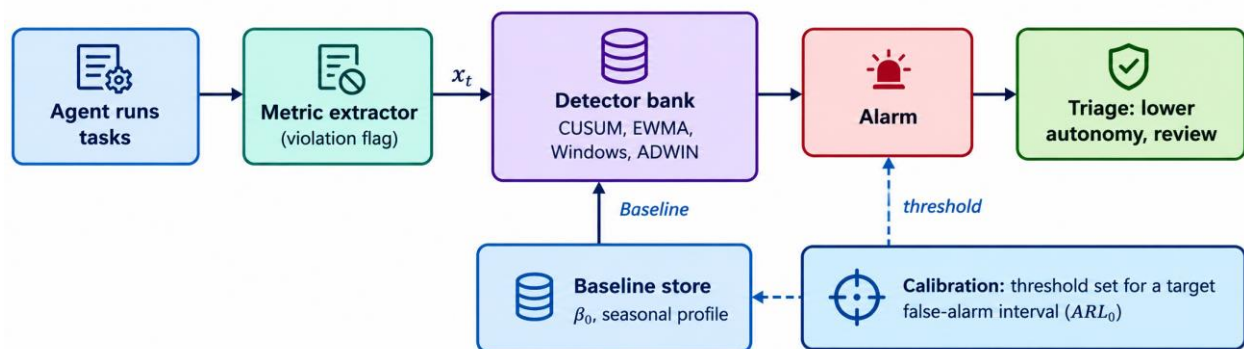


Figure. 1 Monitoring pipeline. Outcomes feed a bank of detectors whose thresholds are calibrated to a target false-alarm interval. An alarm triggers triage, for example lowering autonomy or sending more actions to review. The baseline store holds the estimated normal rate and, if needed, a seasonal profile.

Proposition 2 (Diffusion hazard law). For a reflected Brownian motion with drift μ , variance σ^2 and boundary H , the expected time to hit the boundary from zero is

$$\int_0^{\tau} [1 / ARL(rs - k)] ds = \ln 2$$

Derivation. Eq. (2) is the standard expression for a reflected diffusion. The quasi-stationary assumption is that the walk equilibrates much faster than the drift changes, so that over a short interval the alarm rate is the reciprocal of the ARL at the current drift.

The median delay τ then solves

Eq. (2) is a continuous-time approximation. A discrete walk with jumps overshoots the boundary $\sigma^2 = p_0(1 - p_0)$, so the formula overestimates ARL for the same H . We correct this with one parameter: replace H by an effective boundary H_{eff} chosen so that Eq. (2) at $\mu_s = r_s - k$ gives the

measured ARL_0 of 1,000. Here $H_{\text{eff}} = 3.25$ against the raw $H = 3.55$. No ramp data are used in this correction. Section 6.3 tests the result.

$$ARL(\mu) = [\sigma^2/(2\mu^2)] \cdot [\exp(-2\mu H/\sigma^2) + 2\mu H/\sigma^2 - 1]$$

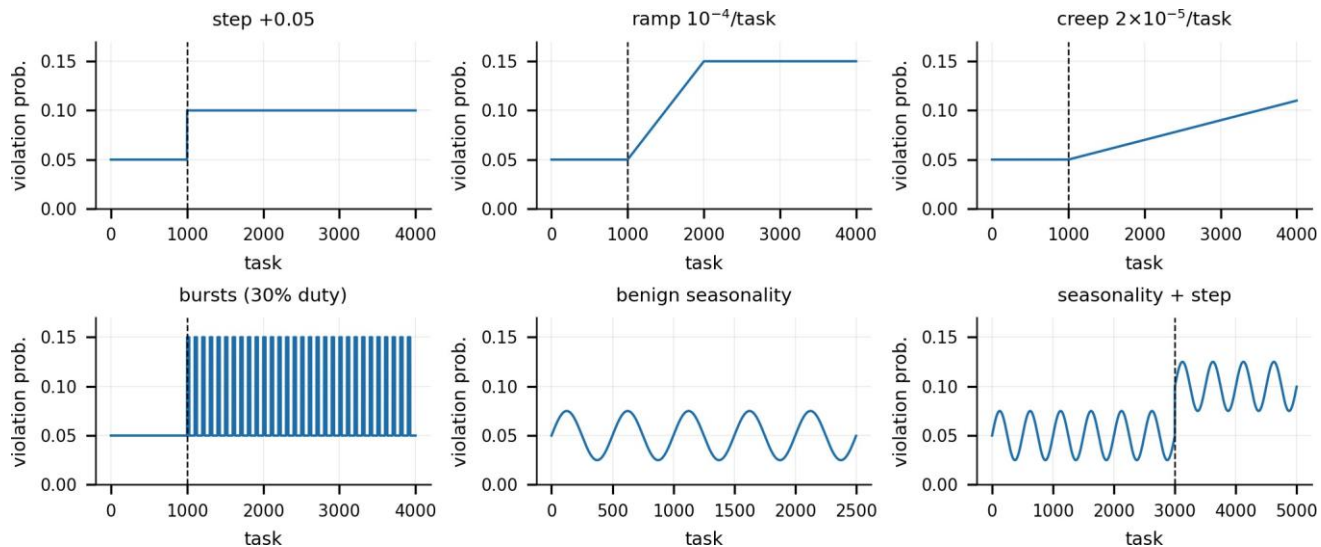


Figure. 2 Drift scenarios. Top row: an abrupt step, a ramp that reaches its final level over 1,000 tasks, and slow creep. Bottom row: bursts that are on 30% of the time, a benign seasonal cycle with 50% amplitude, and the same cycle with a real step at task 3,000.

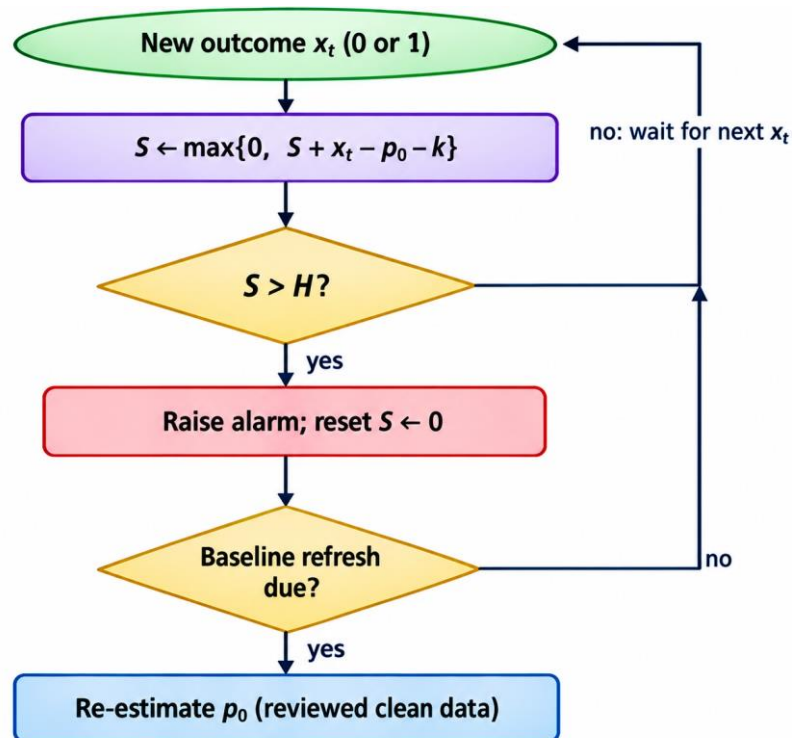


Figure. 3 Per-observation update of the CUSUM detector, including a periodic baseline refresh.

4.2 Baseline Error And Seasonality

Two further effects follow from the same walk. If the baseline is estimated as $\hat{p}_0 = p_0 + \epsilon$, the CUSUM drift

under no change is $-(k + \epsilon)$, so a negative error $\epsilon < 0$ makes the statistic drift upward and shortens ARL_0 .

$$P(\text{no alarm by } \tau) = \exp\{-\int_0^{\tau} [1 / ARL(\mu_s)] ds\}$$

With an estimate from $n_0 = 500$ tasks, ϵ has standard deviation $p_0(1-p_0)/n_0 = 0.0097$. inflation of about 1.2 times for the CUSUM. We measure 1.09 times, so the model is somewhat pessimistic, and the direction is right.

$$\text{ARL}(-k; H_{\text{eff}}) = 1,000; H_{\text{eff}} = 3.25 \text{ (raw } H = 3.55)$$

For benign periodic variation with amplitude a in the true rate $\lim (\mu \rightarrow 0) \text{ ARL}(\mu) = H^2/\sigma^2$, the effective drift is positive during part of each cycle once $a > k$.

4.3 Experimental Setup

Unless stated, $p_0 = 0.05$, drift starts at $t_0 = 1,000$, $a = 0.5p_0 = 0.025 = k$ and we observe 3,000 tasks afterwards. Each scenario uses 1,000 streams. We report the median delay (tasks from t_0 to the first alarm), the share of streams detected within the horizon (always above 100% here), and the false-alarm rate before t_0 , which averaged 0.74 per 1,000 tasks across all scenarios, consistent with the target of 1. Scenarios are a step of +0.02, +0.05 and +0.10, a ramp of 10^{-4} per task, slow creep of $2 \cdot 10^{-5}$ per task, and bursts of +0.10 lasting 30 tasks out of every 100. Medians have standard errors of about 2 to 4 tasks in the step scenarios. Seeds are fixed and all detectors see the same streams.

5. Results and Analysis

5.1. Step changes

Table 2 and Figure 4 give median delays. For a step of +0.05, the CUSUM, CUSUM-LLR, EWMA and ADWIN-style detectors all need about 69 to 75 tasks. They cannot be told apart in practice. The rolling mean needs 79 and the sliding-window z-test 120, which is 1.7 times the CUSUM.

For the larger step of +0.10 the CUSUM family needs only about 30 tasks while the rolling mean needs 44 (1.5 times) and the z-test 87 (2.9 times). For the small step of +0.02, all detectors except the z-test are close, between 180 and 198 tasks, and the z-test needs 246. The CUSUM, EWMA and ADWIN-style results are close because they all approximate the same sequential test when tuned to a shift of about this size. The z-test is slow because its comparison window of 400 tasks holds few violations, so the baseline estimate it compares against is itself noisy.

Step: $\Delta p \in \{0.02, 0.05, 0.10\}$;
ramp: $r = 10^{-4}$;
creep: $r = 2 \times 10^{-5}$

5.2. Speed Against False Alarms

Figure 5 varies the false-alarm interval from 100 to 10,000 tasks, calibrating each detector at each value, and plots the median delay for the +0.05 step. Delay grows roughly with the logarithm of ARL_0 for the CUSUM family, from 19 tasks at $\text{ARL}_0 = 97$ to 149 at 10191. The ADWIN-style detector is

$$\mu = -(k + \epsilon)$$

$$\text{SD}(\epsilon) = \sqrt{[p_0(1-p_0)/n_0]} \approx 0.0097 \text{ (} n_0 = 500)$$

In our setting $a = 0.5p_0 = 0.025$ equals k at 50% amplitude, hazard(s) = $1 / \text{ARL}(\mu_s)$ and false alarms become frequent beyond that. A practical rule is to choose k larger than the largest benign swing, or to model the seasonal profile.

fastest at the shortest intervals but grows faster (27 to 202). The window detectors could not be set to every target.

The rolling mean reaches only 171 at its lowest setting and the z-test 610, because their thresholds move in coarse steps through discrete counts, so we plot them at their achieved values. This is a small point with a practical lesson: windowed counts give you less control over the false-alarm rate than a CUSUM does.

5.3. Ramps And Creep

For the ramp of 10^{-4} per task, all six detectors need about 251 to 290 tasks. By then the violation rate has risen by about 0.027 above baseline. Under slower creep ($2 \cdot 10^{-5}$ per task) the detectors need about 451 tasks, apart from the z-test, which is the fastest at 400. By the time of detection the rate has crept up by about 0.010. The z-test wins here because it compares with a window that is 400 tasks old, which is a moving baseline.

For gradual drift the comparison point is itself drifting, and a window detector can see the difference sooner than a detector tied to a fixed p_0 would, but only when the drift is slow enough that the old window is still close to normal. Table 3 and Figure 6 test the delay laws of Section 4 for the CUSUM against the simulated median over ramp rates from 10^{-5} to 10^{-3} per task.

The first-order law of Eq. (1) overestimates the delay by factors from 1.3 to 9. The diffusion law of Eq. (3) with the corrected boundary stays within 8% (mean error 3%). With the raw boundary H it overestimates by up to about 30%. So the improvement comes both from modelling the noise and from the boundary correction, and the second is a one-parameter fit to the false-alarm interval, not to the ramp data.

5.4. Bursts

When drift is intermittent, with the rate elevated 30% of the time, the CUSUM family and ADWIN-style detectors need 42 to 42 tasks. The rolling mean needs 102 (2.4 times as long) and the z-test 204 (4.8 times). The windowed detectors are slow because a burst of 30 tasks adds only a few violations to a window of 100 or more, and the next quiet stretch dilutes it. Sequential detectors keep a

running total of evidence, so quiet stretches do not erase it as fast. Figure 7 shows example runs for a step and a ramp.

Table. 2 Median detection delay in tasks (1,000 streams; false-alarm interval 1,000). Bold marks the fastest in each row. “Step 0.10” means the violation rate rises by 0.10.

Scenario	CUSUM	CUSUM-LLR	EWMA	Rolling	Window z	ADWIN-style
step 0.02	180	179	183	198	246	179
step 0.05	69	69	70	79	120	75
step 0.10	30	30	31	44	87	31
ramp (1e-4)	251	248	248	260	290	269
creep (2e-5)	451	450	445	464	400	456
bursts 0.10	42	44	48	102	204	42

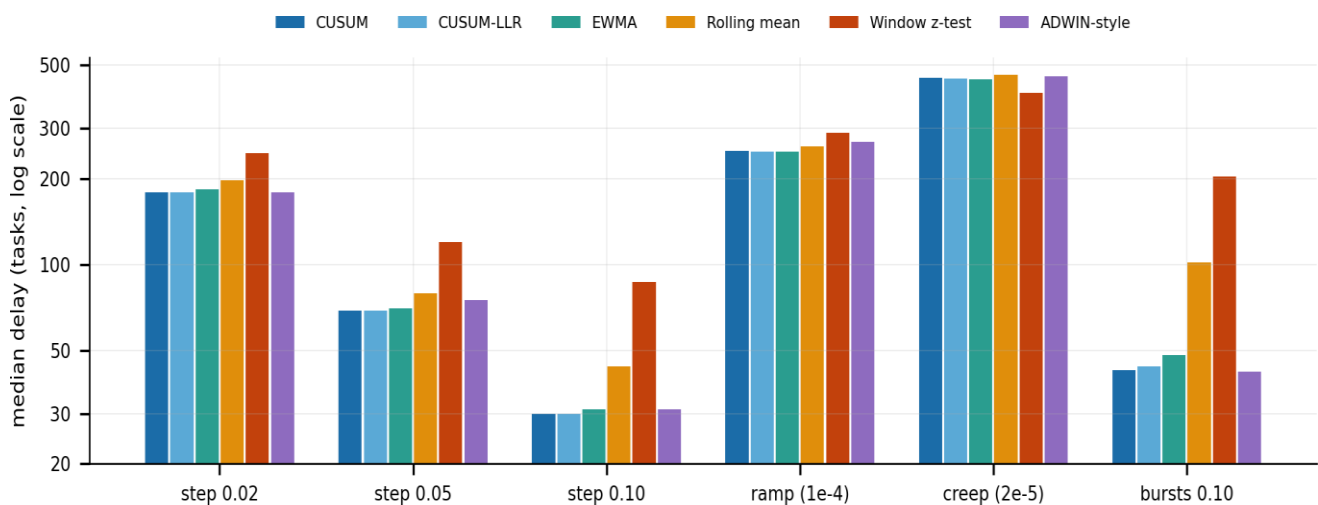


Figure. 4 Median delay by scenario and detector (log scale). The CUSUM family, EWMA and ADWIN-style detectors are close together. The windowed test and the rolling mean are slower on abrupt and bursty drift.

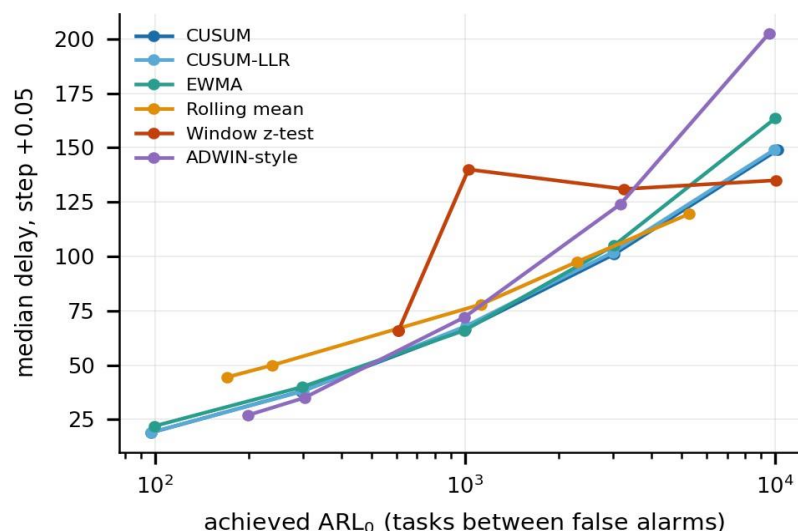


Figure 5: Median delay for a step of +0.05 against achieved false-alarm interval. Curves for the windowed detectors start at higher intervals because their thresholds cannot reach the smallest targets.

5.5. An Estimated Baseline

Real deployments do not know p_0 . We estimated it from the first 500 tasks of each stream and used the same calibrated thresholds as before (Figure 8). The detectors that use the

baseline (CUSUM, CUSUM-LLR, EWMA, rolling mean) raised more false alarms: the CUSUM-LLR

Table 3: CUSUM delay for ramp drift (median, tasks; 600 streams). First-order law Eq. (1); diffusion law with raw and corrected boundary.

Rate	First-order	Diffusion, H	Diffusion, H_{eff}	Simulated
1e-05	5138	722	561	564
2e-05	2635	594	483	460
5e-05	1126	415	358	350
1e-04	615	298	266	258
2e-04	351	207	190	192
5e-04	179	124	117	114
1e-03	113	84	80	87

went from 0.98 to 1.35 per 1,000 tasks, the EWMA from 0.96 to 1.39, and the rolling mean from 0.87 to 1.31. The CUSUM itself went from 0.99 to 1.07. Detectors that compare two windows, the z-test and the ADWIN-style detector, need no baseline and were unaffected. Delays hardly changed. For the CUSUM, the median was 70 with the true baseline and 68 with the estimate. The cost of a baseline estimated from 500 tasks is therefore a modest rise in false alarms, up to about 1.39 times the target, with no loss in speed. Longer baselines or periodic refresh would reduce it.

5.6. Benign seasonality

Agent traffic often varies by time of day, and with it the violation rate. We added a seasonal cycle with period 500 tasks to the stream and varied its amplitude as a fraction of p_0 (Figure 9a). With no seasonality, false alarms are close to the target. At 50% amplitude they rise to between 1.4 and 2.1 per 1,000 tasks for the plain detectors, and at 100% amplitude to 3.1 for the CUSUM, 3.3 for the EWMA and 3.9 for the ADWIN-style detector.

The z-test does best, at 1.9, since its comparison window is long relative to the cycle. A CUSUM that subtracts a seasonal profile learned from the first four cycles keeps the rate at 1.7 at 100% amplitude, roughly halving the excess. It is not free of cost: with no seasonality at all, this version raises 1.3 per 1,000 because the profile is itself noisy. Detection delay with a real step under 100% seasonality is 52 for the CUSUM and 58 for the seasonal version, so correcting for seasonality costs almost nothing in speed.

5.7. Cost Of Running The Detectors

Table 4 gives the time per observation of plain Python implementations on the sandbox machine (one core, no optimisation, 300,000 observations). The CUSUM, EWMA and rolling-mean detectors take about $0.11 \mu\text{s}$ per observation, the z-test $0.62 \mu\text{s}$, and the ADWIN-style detector $1.93 \mu\text{s}$, about 18 times as much because it checks five window splits. Even the slowest handles about half a million observations per second. Compute is not a reason to choose among these detectors.

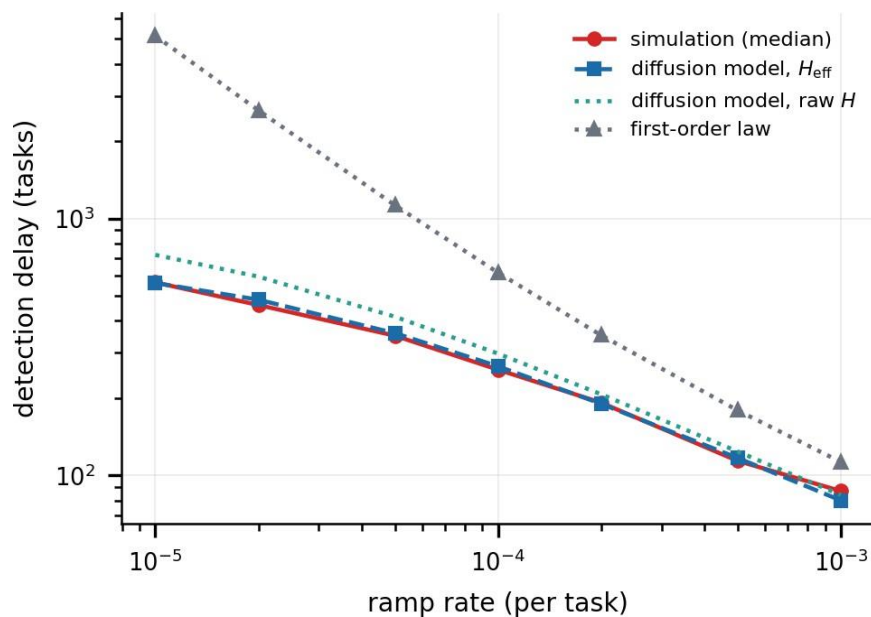
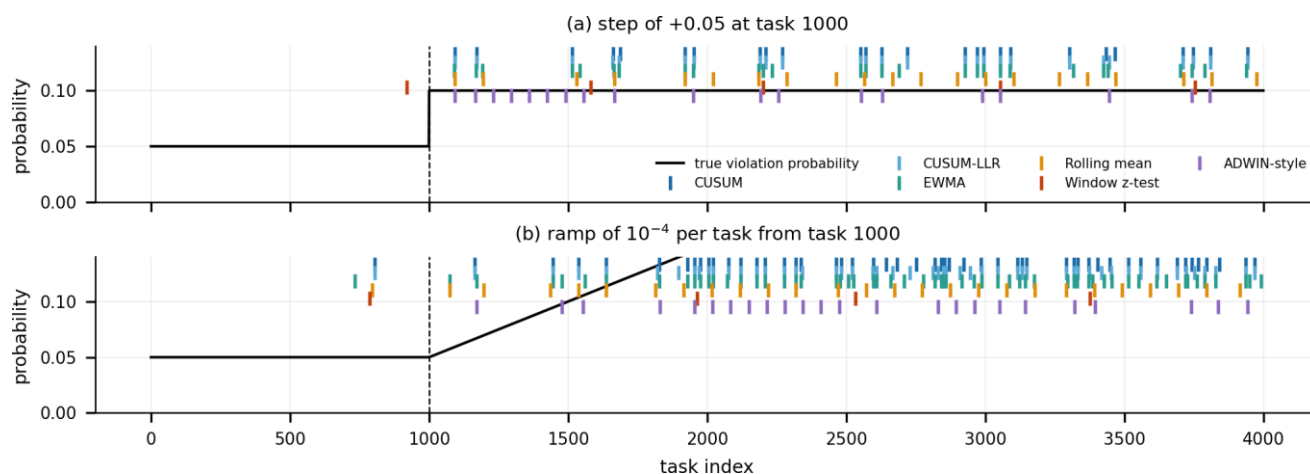
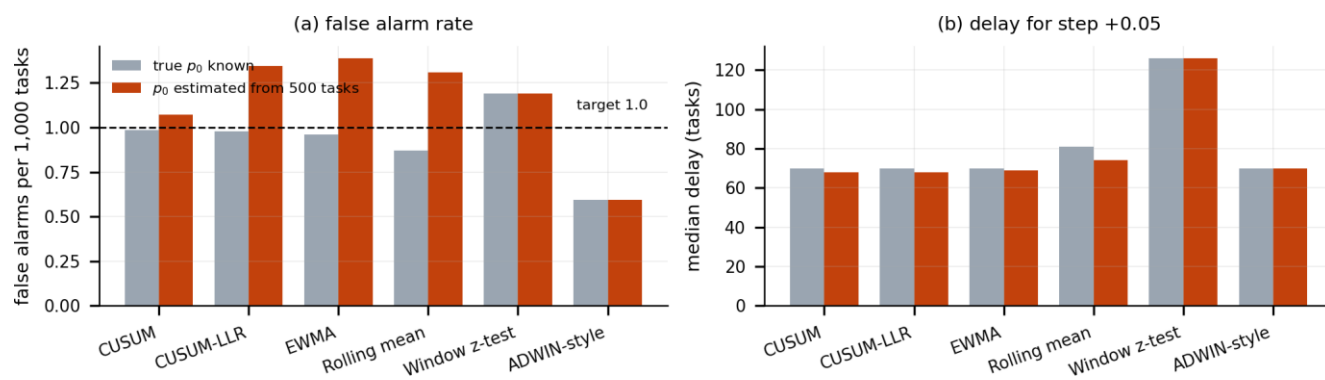
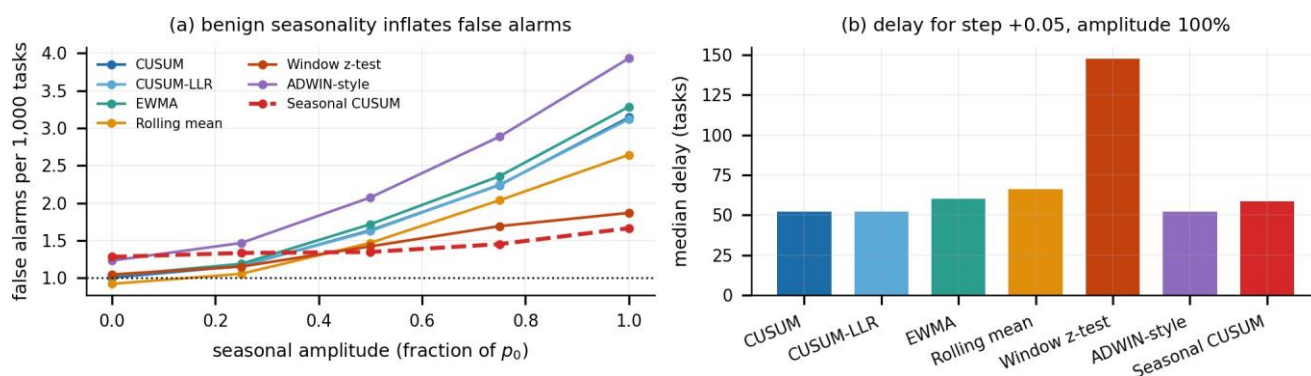


Figure. 6 Detection delay for ramp drift: simulation against the first-order law and the diffusion law (log-log). The first-order law is far off for slow ramps.

Table 4: Per-observation cost of scalar Python implementations.

Detector	μs per observation	Millions per second
CUSUM	0.11	9.2
CUSUM-LLR	0.10	9.8
EWMA	0.08	11.9
Rolling mean	0.11	9.5
Window z-test	0.62	1.6
ADWIN-style	1.93	0.5

**Figure 7** One example run of each drift pattern. Ticks mark the alarms each detector raises (rows). After a real change, alarms keep recurring because each detector resets and then sees the new level as an ongoing departure from the old baseline.**Figure 8:** Effect of estimating the baseline from 500 tasks. (a) False alarms per 1,000 tasks; the target is 1. (b) Median delay for a +0.05 step. Window comparison detectors need no baseline.**Figure 9:** (a) False alarms per 1,000 tasks as the seasonal amplitude grows. The dotted line is the target.

(b) Median delay for a +0.05 step when seasonality has 100% amplitude.

5.8. Discussion and Limitations

What to use. For a stream of violation flags with small, sustained or bursty drift, a CUSUM or EWMA tuned to the smallest change you care about is as fast as anything we tried, is easy to calibrate, and handles bursts well. A windowed comparison is the better choice when the baseline is unknown or drifts slowly, since it needs no baseline estimate, and its false-alarm rate is more stable under seasonality. Windows should be long enough to hold a handful of violations.

Simulation, not logs. Our streams are independent Bernoulli sequences. Real agent outcomes are autocorrelated, come in clusters, depend on task type, and are measured with error. We expect clustering to raise false alarms for every detector, and we did not test it. The detectors' calibration would need to be done on real null data, or on resampled blocks.

One metric. We monitored a single violation rate. In practice there are many metrics (latency, tool errors, refusal rate, score distributions), and monitoring all of them multiplies false alarms. Multiple-testing corrections and combined statistics are outside this paper.

The theory is approximate. The diffusion law fits well in our setting, with $p_0 = 5\%$ and a CUSUM. It relies on a continuous approximation of a discrete walk, on a calibrated effective boundary, and on quasi-stationarity. We have not tested it for other base rates, other detectors or faster drift beyond the range shown. The baseline-error model overstated false-alarm inflation (1.2 against 1.09).

What happens after an alarm. We did not model the response. An alarm that triggers review costs human attention (see Paper 2 of this series on fatigue) and a false one wastes it, which is why we set the false-alarm interval explicitly. The right ARL_0 depends on the cost of an alarm and the cost of delay, and we did not study that trade-off.

6. Conclusion and Future Work

At matched false-alarm rates, the standard sequential detectors (CUSUM, EWMA and an adaptive window) noticed a step from 5% to 10% in about 69 tasks, and the plain windowed tests needed up to 1.7 times as many. On bursty drift the gap was larger. The classical delay formula was a poor guide at a false-alarm interval of 1,000 tasks, and a diffusion-based formula with one calibrated boundary predicted ramp delay to within 8%. Estimating the baseline and ignoring seasonality both raised false alarms, by up to a factor of 1.39 and about three, and a seasonal profile recovered part of the loss. The next steps

are to test the detectors on real agent logs, to handle several metrics at once, and to choose the false-alarm interval from the cost of an alarm.

References

- [1]. E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.
- [2]. G. Lorden. Procedures for reacting to a change in distribution.
- [3]. *Annals of Mathematical Statistics*, 42(6):1897–1908, 1971.
- [4]. G. V. Moustakides. Optimal stopping times for detecting changes in distributions. *Annals of Statistics*, 14(4):1379–1387, 1986.
- [5]. T. L. Lai. Information bounds and quick detection of parameter changes in stochastic systems. *IEEE Trans. Information Theory*, 44(7):2917–2929, 1998.
- [6]. S. W. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1(3):239–250, 1959.
- [7]. M. Basseville, I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, 1993.
- [8]. D. Siegmund. *Sequential Analysis: Tests and Confidence Intervals*. Springer, 1985.
- [9]. J. Gama et al. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44, 2014.
- [10]. J. Lu et al. Learning under concept drift: a review. *IEEE Trans. Knowledge and Data Engineering*, 31(12):2346–2363, 2019.
- [11]. Bifet, R. Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM Int. Conf. Data Mining*, 2007.

How To Cite:

Appiah Prince , Vijayalakshmi Kakulapati , " Noticing the Slide: Calibrated Drift Detection for Deployed Agents " , *International Journal of Computational Sciences and Engineering Research* , Volume 3, Issue 4 , pp 13 – 22 , October 2026.

CrossRef DOI: <https://doi.org/10.63328/ijcser-v3ri4p2>

Parameters

Baseline $p_0 = 0.05$. Calibration: 800 null streams of 5,000 tasks, bisection with 13 iterations, target $ARL_0 = 1,000$; verification on 1,500 fresh streams of 6,000 tasks. Evaluation: onset at task 1,000, 3,000 tasks observed, 1,000 streams per scenario. Estimated baseline: mean of first 500 tasks, onset at 1,500. Seasonality: period 500, profile

estimated from the first four cycles (2,000 tasks), onset at 3,000. Ramp-law test: onset 1,000, 6,000 tasks observed, 600 streams. Timing: 300,000 observations, plain Python. Code and figures are in the supplementary archive. All figures were generated from code at 300 dpi and printed at their stored width.

Declaration

Conflicts of Interest: The authors declare no conflict of interest.

Author Contribution: All authors wrote the main manuscript text and also consent to the submission.

Ethical approval: Not applicable.

Consent to Participate: All authors consent to participate.

Funding: Not applicable, and No funding was received

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Personal Statement: We declare with our best of knowledge that this research work is purely Original Work and No third party material used in this article drafting. If any such kind material found in further online publication, we are responsible only for any judicial and copyright issues.

Generative AI Statement: The authors declare that generative AI was not used in the preparation or creation of this manuscript. The alternative text (alt text) accompanying the figures in this article was generated by Frontiers with the assistance of artificial intelligence. Reasonable efforts have been made to ensure the accuracy and appropriateness of the generated alt text, including review by the authors wherever possible. If you identify any inaccuracies or issues, please contact the editorial team.

Publisher's Note: The statements, opinions, and claims presented in this article are exclusively those of the authors and do not necessarily reflect the views of their affiliated institutions, the publisher, editors, or reviewers. Any products discussed or evaluated, as well as any claims made by their manufacturers, are not warranted, approved, or endorsed by the publisher.

Acknowledgements

We thank everyone who inspired our work