# Two-Ticker Output Summary

Amit Amlani code + ChatGPT additions (shape-only ticker, plateaus, and discrepancy reporting)

Date: January 09, 2026

This document describes what the existing (legacy) ticker logic does and what the additional "ChatGPT" logic does, including how the two outputs complement each other when you run them in parallel. It focuses on what you see in the console output: summary ranges, ticker strings, plateau reporting, and the discrepancy ("consistent / inconsistent") analysis.

## At a glance

- Legacy ticker ("Amit Amlani Tested TICKER") is magnitude/reset oriented: it is sensitive to jumps or out-of-step moves and can split or shorten chains, which can be useful as an event/regime-change signal.
- ChatGPT ticker ("shape-only") is a reference ticker: it counts local step patterns consistently, adds explicit plateau detection, and provides a structured breakdown of plateau locations and sizes.
- A discrepancy between the two tickers is treated as signal: either plateau-related (expected) or a true behavioral difference (typically magnitude/count differences in A/D chains).

## Core concepts and vocabulary

| Term | Meaning in your outputs |
|---|---|
| nums | Input float array being analyzed. |
| Summary range | Human-readable representation of the sequence as standalones and step-ranges (e.g., 19.6->20.0). |
| Ticker | Compact signature of behavior: S for standalone, A for ascending chain, D for descending chain, and '-' as a transition separator. |
| Step size | The increment used to qualify A/D steps (e.g., +0.1 for A, -0.1 for D). Moves that are not exactly +/- step size are treated as S (standalone) in the step-based logic. |
| Plateau | A run of equal consecutive values (e.g., 1.65, 1.65). Plateau detection is value-equality based and does not depend on step size. |
| k | Plateau start index in nums (0-indexed). |

## What the legacy ticker does (magnitude/reset)

The legacy path is your original implementation that builds (1) your summary ranges and (2) your "Amit Amlani Tested TICKER". It encodes behavior in a compact way and is intentionally sensitive to discontinuities. When the input includes big jumps or steps that do not match the configured step size, the legacy ticker can treat those points as boundaries and restart or split counting. This can produce chain magnitudes that differ from the shape-only ticker (for example A(4) versus A(5)).

**Where the legacy output relies on ChatGPT-added code:** the legacy logic remains the source of the original summary range and "Amit" ticker string, but it now benefits from a few integrated utilities: (a) S-run compression formatting (S(n)) applied consistently to keep long outputs readable, (b) the side-by-side discrepancy report that parses both ticker strings and summarizes mismatches, and (c) optional mismatch tagging/subtotals that use the ChatGPT plateau list to label plateau-related S-run differences. These additions do not change the legacy decision rules; they augment how results are reported.

### Why this can be useful

- Treats jumps/out-of-band moves as regime boundaries (good for detecting resets, feed stitching, or event shocks).
- Can be used as an "event ticker" when you care about discontinuities more than micro-step fidelity.

## What the ChatGPT additions do (shape-only ticker, plateaus, and reporting)

A separate shape-only ticker builder was added to compute a reference view of the local step pattern. This code path does not try to modify or "correct" the legacy ticker. Instead, it produces a second ticker that you can compare against the legacy output to find where and why the legacy behavior differs.

**Where the ChatGPT output relies on your legacy code:** the ChatGPT ticker runs on the same input arrays (nums) and uses the same step-size definition that your code uses to decide what counts as A and D. It is invoked from within the existing reporting flow so the two outputs appear together in the console. The discrepancy checker also treats the legacy ticker string as the baseline to parse/align against, so your original formatting conventions remain the reference.

### Key behaviors

- Builds CHATGPT SUMMARY RANGE and CHATGPT TICKER (shape-only).
- Detects plateaus (equal consecutive values) and marks plateau start in the ChatGPT ticker with '*'.
- Prints plateau details: plateau index k, size, value, and a local context label (e.g., 'Between S and S').
- Computes plateau subtotals by label, with both plateau count and occurrences (sum of plateau sizes).

### Plateau reporting details

Plateau detection is based on value equality only: if nums[i] == nums[i+1] (within float equality used by the code), a plateau run begins. The step size used for A/D does not affect whether a plateau is detected. However, the plateau context label (the 'Between ... and ...' classification) is derived from looking one step before and one step after the plateau using the same step-size rule that defines A and D.

### Interpreting subtotals

In the subtotals table, each label prints two numbers: (1) count = how many plateaus matched that label, and (2) occurrences = how many repeated values occurred in total for that label (sum of plateau sizes). So a single plateau of size 2 contributes count=1 and occurrences=2.

## How the two tickers complement each other

Running both tickers side-by-side gives you two lenses on the same data: the legacy output highlights discontinuities and resets (macro events), while the shape-only ticker preserves micro-step fidelity (local shape). When they differ, that difference is actionable information: it tells you exactly where your legacy rules treat the stream as a new regime or boundary.

- Debugging: you can spot where the legacy ticker split a chain early by comparing magnitudes and index ranges.
- Monitoring/signaling: discrepancies often correlate with resets, outliers, feed changes, rounding/quantization changes, or step-size violations.
- Explainability: the mismatch report provides segment indices and element index ranges so you can jump directly to the raw nums window.

# Discrepancy reporting (consistent / inconsistent)

After both tickers are produced, a comparison step runs. For reliable alignment, the comparison normalizes the ticker strings for comparison only: it ignores plateau '*' markers and '-' separators and merges adjacent same-type tokens so that standalone runs align (e.g., S + S(2) becomes S(3)). This prevents a single plateau marker from causing a knock-on misalignment across the remainder of the chain.

## Mismatch tags

| Tag | Meaning |
|---|---|
| (plateau-related) | Mismatch overlaps a plateau range and involves S tokens. Usually expected due to plateau marker splitting standalone runs. |
| (true behavioral differences) | Mismatch not explained by plateaus; typically an A/D magnitude difference or segment-type difference. |
| (blank/unknown) | Fallback category (e.g., missing token on one side). Often indicates an alignment/coverage edge case in the comparison layer. |

# Examples

## Example A: magnitude mismatch after a discontinuity

nums: 35.3, 35.2, 35.1, 85.6, 85.7, 19.6, 19.7, 19.8, 19.9, 20.0, 19.9, 19.8. Legacy ticker may split counting around the large jump, producing A(4), while the shape-only ticker continues the local +step run and produces A(5). This is treated as a true behavioral difference.

## Example B: plateau splitting without behavioral change

If the data contains a plateau inside an otherwise standalone run, the ChatGPT ticker inserts '*' at the plateau start (and prints plateau details). The comparison logic merges adjacent S runs during comparison, so the remainder of the chain stays aligned and downstream A/D segments compare correctly. S-run mismatches in such windows are tagged plateau-related.

# How to use this during testing

- Run your full suite of test arrays at the top of the file.
- When you see 'Inconsistent ticker', scan the mismatch subtotals first: plateau-related mismatches are usually expected; true behavioral differences are where you investigate magnitude/reset behavior.
- Use the printed (start - end) index ranges to locate the exact window in nums that caused the mismatch.
- If a case surprises you, double-check the step delta (for example 1.65 -> 1.64 is -0.01, not -0.1).