1. Experimental Results

Case 1: 100,000 ticks

Total Orders: 70,820

Average Tick-to-Trade Latency: ~69.9 ms

Maximum Latency: ~113.9 ms

Signal Contributions:

Signal 1 (Breakout): 1,381

Signal 2 (Mean Deviation): 66,750

Signal 3 (Momentum): 16,711

Total Runtime: 131 ms

2. Write-Up

1) Which signal triggered the most orders?
From the performance reports, Signal 2 (Mean Deviation) clearly triggered the most orders. With 100k ticks it contributed over 66k orders, far more than Signal 3 (Momentum ~16k) and Signal 1 (Breakout ~1.3k). This shows that small deviations from the short-term average are the dominant driver of trading activity in this setup.

2) What could you optimize further?
In terms of optimization, the main bottlenecks are the use of erase(begin()) on vectors for price history and recomputing averages/standard deviations from scratch. Both introduce unnecessary O(n) work per tick. Replacing them with a fixed-size ring buffer and incremental statistics would cut per-tick cost to O(1). Reducing frequent calls to chrono::now() and minimizing memory reallocations would also improve latency.

3) How would your code behave with 10x more data?

Case 2: 1,000,000 ticks

Total Orders: 707,379

Average Tick-to-Trade Latency: ~557.3 ms

Maximum Latency: ~1,045.5 ms

Signal Contributions:

Signal 1 (Breakout): 14,070

Signal 2 (Mean Deviation): 666,537

Signal 3 (Momentum): 166,796

Total Runtime: 1161 ms

When scaling from 100k to 1M ticks, runtime grew from ~131 ms to ~1161 ms, roughly linear. However, average tick-to-trade latency worsened from ~70 ms to ~560 ms, with maxima over 1 second. This indicates that while throughput scales, latency degrades quickly without structural optimizations.