Simulation of Automated Market Makers for the Analysis of Crypto Trends (in Chaotic Scenarios)

Alessandro Tricca – alessandro.tricca@studio.unibo.it
Jonathan Ted Benson Cerullo Uyi – jonathan.cerullouyi@studio.unibo.it

Contents

1 Introduction

Automated Market Makers (AMMs) have revolutionized decentralized finance (DeFi) by enabling permissionless token trading without relying on traditional order books or centralized intermediaries. Unlike conventional exchanges that match buyers and sellers, AMMs use mathematical formulas to determine prices algorithmically based on pool reserves.

However, different AMM designs exhibit vastly different behaviors under stress. The choice of pricing formula directly impacts slippage, liquidity efficiency, and resilience during market chaos. In this project, we implement and compare three fundamental AMM models: **Uniswap v2 (Constant Product)**, **Constant Sum** and **Curve StableSwap**. Furthermore, we include several types of economic agents that interact with the AMM pools: **Retail Traders**, an **Arbitrageur Bot**, a **Panic Liquidity Provider** and a **Whale Trader**.

Through these interactions, we observe emergent phenomena such as price divergence, arbitrage corrections, slippage escalation, and liquidity crises. The modular architecture allows us to isolate variables and measure each AMM's performance across standardized stress tests.

1.1 Research Questions

This work addresses three key questions:

1. **Stability:** Which AMM design maintains price accuracy under extreme volatility?
2. **Resilience:** How do different formulas respond to liquidity shocks and panic withdrawals?
3. **Efficiency:** What are the trade-offs between slippage, capital efficiency, and robustness?

Our findings demonstrate that there is no universal "best" AMM: the optimal choice depends critically on the correlation between traded assets. Constant Product excels for volatile pairs (ETH/USDC), while Constant Sum and Curve are designed for stablecoin pairs (USDC/USDT). Mismatched designs lead to catastrophic failures, as we document in Section 5.

Overall, this work provides a clear and flexible tool for understanding AMM behavior under stress, highlighting both the innovations and inherent limitations of algorithmic market making in decentralized finance.

2 Simulator Architecture

The simulator is structured around modular Python components, each representing a different aspect of an AMM ecosystem.

2.1 AMM Engine

The simulator implements three distinct AMM models to enable comparative analysis:

2.1.1 Uniswap v2 - Constant Product Market Maker (CPMM)

$$x \cdot y = k$$

The CPMM maintains a hyperbolic price curve. As reserves become imbalanced, prices adjust exponentially. This provides infinite liquidity (you can never completely drain the pool) but causes high slippage on large trades.

2.1.2 Constant Sum Market Maker (CSMM)

$$x + y = k$$

The CSMM offers near 1:1 swap rates, making it ideal for assets with equal value (e.g., USDC/USDT). However, it's **unstable** for volatile pairs like ETH/USDC, as the pool can be completely drained if prices diverge significantly.

2.1.3 Curve StableSwap (Hybrid Model)

$$k = (x \cdot y)^{\alpha} \cdot (x + y)^{1-\alpha}, \quad \alpha = \frac{A}{A+1}$$

Curve blends CPMM and CSMM behavior using an amplification parameter A. Near balanced reserves (1:1 ratio), it acts like CSMM (low slippage). Far from balance, it converges to CPMM (high slippage).

Note: Our implementation uses a simplified explicit formula for analysis purposes.

2.2 Trader Agents (trader.py)

Retail traders model typical user behavior by performing random buy/sell actions based on their available wallet balances. At each simulation step, a trader may decide to do nothing (Hold) with a certain probability, or instead perform a buy or sell action.
The framework also includes a SmartTrader variant capable of avoiding trades with excessive slippage. These agents introduce trading flow into the AMM, simulating realistic user interaction patterns.

2.3 Advanced Agents (agents.py)

To simulate a realistic market environment, the project implements varying types of economic agents.

2.3.1 The Arbitrageur

The Arbitrageur acts as the market's stabilizer. It constantly monitors the spread between the AMM price and the external market price.

- **Logic:** The agent tracks the difference between the AMM price and the market price. If this gap exceeds a specific profit threshold (defaulting to 0.5%), the agent executes a swap. It buys the undervalued token and sells the overvalued one to close the gap.
- **Effect:** This activity keeps the AMM price aligned with the real world, ensuring the pool reflects the true market value.

2.3.2 Panic Liquidity Provider (PanicLP)

The PanicLP class introduces fragility to the system. Unlike passive LPs who hold through all conditions, this agent monitors market volatility.

- **Stress Reaction:** If volatility exceeds a panic_threshold (e.g., 8%), the agent withdraws liquidity (burns LP tokens).
- **Effect:** This simulates a "bank run" scenario. When big investors pull their money out during a crash, the pool reduces rapidly. This makes trading harder and more expensive for everyone else, potentially causing the market to destabilize further.

2.3.3 The Whale Trader

The Whale Trader represents a wealthy participant capable of moving the market with a single action. Unlike retail traders who make small swaps, the "Whale" executes massive trades that significantly shift the pool's reserves.

- **How it works:** This agent performs specialized "pump" and "dump" operations. It can suddenly buy a huge amount of tokens (pump) or sell a large portion of its holdings (dump) in one go.
- **Effect:** These massive trades cause extreme price changes and high slippage. This helps us stress-test the AMM to see if it remains functional during market manipulation or chaotic events.

3 Simulation Dynamics

The core dynamics of the project are managed by two critical components: the Simulation Engine (run_simulation.py), which coordinates the interaction between agents and the AMM over discrete time steps, and the Interactive Interface (main.py) which allows the user of real-time control over the market.

3.1 The Simulation Class (run_simulation.py)

The Simulation class runs the virtual world. Instead of running continuously, it breaks time into "steps," allowing us to see exactly what happens moment by moment.
When the simulation starts, it builds the market, sets the initial price, and gives funds to all the agents.

The Loop: In every single step, the system follows a strict order:

1. **Price Change:** The external market price moves up or down randomly.
2. **Action:** Agents see the new price and react—Arbitrageurs fix gaps, traders swap tokens, and LPs panic if needed.
3. **Recording:** Crucially, the system saves a snapshot of everything (prices, reserves, and fees) after every action, enabling post-simulation analysis.

3.2 The Main Interface (main.py)

While the Simulation class handles the logic, the main.py module serves as the Command and Control Center. It provides a Command Line Interface (CLI) that gives the user full control via a simple text menu.
Unlike static simulations that run from start to finish without interference, the Main Interface allows the user to pause execution and inject specific market events. The user can manually trigger a Market Shock (e.g., crashing the external price by 30%) or force a Whale Event (pump/dump) to observe the immediate impact on the AMM's liquidity.

4 Experimental Design

4.1 Methodology

We designed four distinct stress-testing scenarios to evaluate AMM resilience:

4.1.1 Baseline Scenario

Normal market conditions with moderate volatility ($\pm 2\%$ per step). This serves as a control group to measure AMM performance under typical trading activity.

4.1.2 Flash Crash

A sudden 50% price drop at step 250/500, simulating a black swan event like the May 2021 crypto crash. Tests AMM ability to maintain liquidity during extreme panic.

4.1.3 Whale Manipulation

Sequential pump-and-dump attacks:

- Step 200: Whale dumps 50% of ETH holdings
- Step 400: Whale pumps with 50% of USDC holdings

Evaluates AMM resistance to market manipulation.

4.1.4 High Volatility

Sustained extreme volatility (±10% per step) over 500 steps. Simulates prolonged market chaos like the 2022 Terra/Luna collapse.

4.2 Metrics

We track the following key performance indicators:

- **Price Gap:** Divergence between AMM and market price
- **Price Stability:** Standard deviation of AMM price
- **Arbitrageur Profit:** Total gains from price corrections
- **LP Panic Events:** Frequency of liquidity withdrawals
- **k Growth:** Fee accumulation rate (pool health)

4.3 Implementation

Each experiment was repeated 10 times with different random seeds (seed = 42 + offset) to ensure statistical validity. We report mean ± standard deviation across runs.

5 Results

5.1 Comparative Analysis

Figure 1 provides a comprehensive view of how all three AMM implementations perform across the four experimental scenarios. The following subsections analyze Uniswap v2 (CPMM) in detail, as it demonstrates the most robust performance for volatile asset pairs. Comparative analysis with Constant Sum and Curve is provided in Sections 1 and 2.

5.1.1 Baseline Performance

Under normal conditions:

- Average price gap: 5.22 ± 0.36 USDC (0.26%)
- Arbitrageur profit: -$1,959 ± 2,289 (negative due to gas costs)
- Zero LP panic events
- Price stability: 127.27 (standard deviation of AMM price)

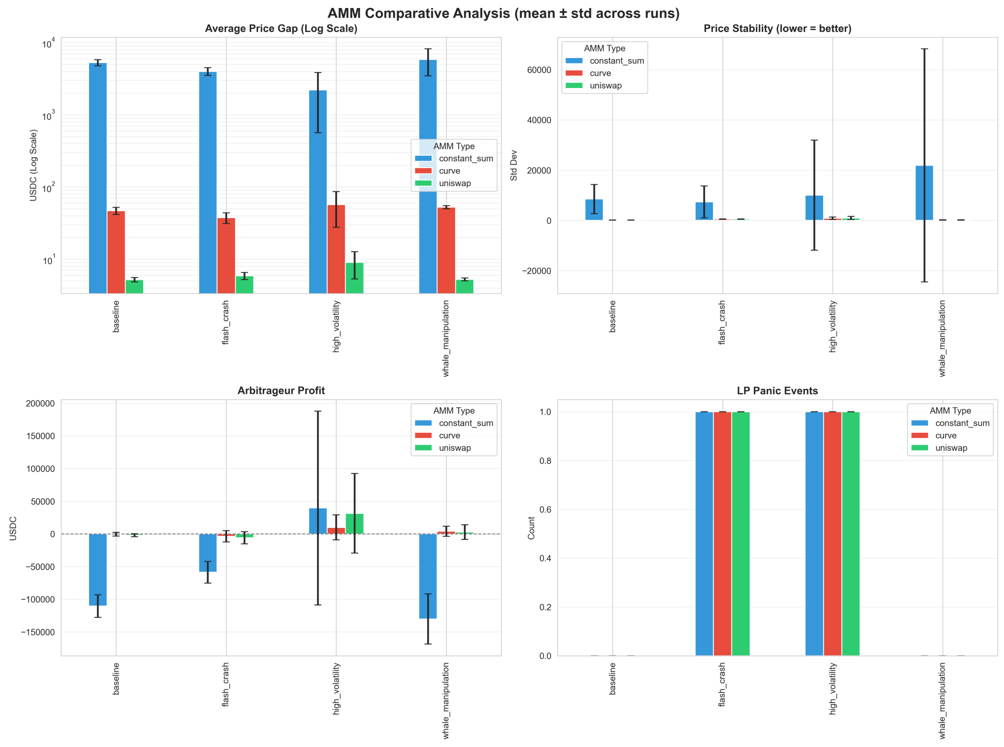

Figure 1: Comparative Performance Metrics: Uniswap (blue), Constant Sum (red), and Curve (green) Across All Experimental Scenarios. Note the logarithmic scale in Price Gap subplot, highlighting Constant Sum's orders-of-magnitude divergence.

5.1.2 Note on Arbitrageur Profitability

The negative arbitrage profits observed in most scenarios require clarification. In our simulation, each arbitrage transaction incurs a fixed cost representing gas fees and slippage. When price gaps are small (as in the Baseline and Flash Crash scenarios), these transaction costs exceed the profit from closing the gap.

Key insights:

- **Baseline/Flash Crash:** Small price gaps (<6 USDC) make arbitrage unprofitable after accounting for transaction costs
- **High Volatility:** Large, sustained price gaps ($9.01 USDC) enable profitable arbitrage ($31,506 profit)
- **Real-world parallel:** This reflects actual DeFi dynamics where only sophisticated MEV (Maximal Extractable Value) bots with optimized gas strategies can profit from small arbitrage opportunities

This economic reality acts as a natural damper on arbitrage activity, allowing temporary price inefficiencies to persist - a phenomenon we observe in our results.

5.1.3 Flash Crash Impact

The 50% price drop triggered:

- Average price gap: 5.88 ± 0.65 USDC (12% increase vs. baseline)
- LP panic withdrawal occurred (1.0 events on average)
- Arbitrageur profit: -$5,960 ± 9,236 (losses amplified by volatility)
- Price stability degraded to 530.18 (4× baseline volatility)

5.1.4 Whale Manipulation

Sequential pump-and-dump attacks showed:

- Average price gap: 5.25 ± 0.25 USDC (comparable to baseline)
- Arbitrageur profit: $2,636 ± 11,250 (positive but highly variable)
- Zero LP panic events (market resilient to manipulation)
- Price stability: 217.80 (71% higher than baseline)

5.1.5 High Volatility

Sustained extreme volatility revealed:

- Average price gap: 9.01 ± 3.71 USDC (73% higher than baseline)
- Arbitrageur profit: $31,506 ± 60,925 (highest across all scenarios)
- LP panic withdrawal occurred (1.0 events)
- Price stability: 957.78 (7.5× baseline—most chaotic scenario)

5.2 Statistical Significance

Table 1 reports mean ± std across 10 runs.

5.3 Performance Heatmap Analysis

Figure 2 provides a heatmap visualization of key performance metrics across all experimental scenarios. The color gradients reveal clear patterns:

- **Price Gap:** Constant Sum exhibits catastrophic price divergence (thousands of USDC) across all scenarios (visible as dark red cells)
- **Stability:** Curve and Uniswap show comparable stability (green/yellow), while Constant Sum fails completely (red)
- **Arbitrage Profit:** Only High Volatility scenarios enable positive arbitrage (green cells), confirming our findings in Section 5.1.2
- **LP Panics:** Triggered uniformly in Flash Crash and High Volatility scenarios across all AMM types

The heatmap format makes the performance gap between AMM designs immediately apparent: Constant Sum's red-dominated column stands in stark contrast to Uniswap's balanced profile.

Metric	Baseline	Flash Crash	Whale	High Vol
Uniswap v2 (CPMM)				
Price Gap (USDC)	5.22 ± 0.36	5.88 ± 0.65	5.25 ± 0.25	9.01 ± 3.71
Arb Profit ($)	-1,959 ± 2,289	-5,960 ± 9,236	2,636 ± 11,250	31,506 ± 60,925
Stability	127.27	530.18	217.80	957.78
LP Panics	0.0	1.0	0.0	1.0
Constant Sum (CSMM)				
Price Gap (USDC)	5,295 ± 507	3,999 ± 490	5,836 ± 2,371	2,207 ± 1,643
Arb Profit ($)	-110,444 ± 17,356	-58,766 ± 16,444	-130,234 ± 38,412	39,594 ± 148,277
Stability	8,531.99	7,386.88	21,939.10	10,050.25
LP Panics	0.0	1.0	0.0	1.0
Curve StableSwap				
Price Gap (USDC)	46.96 ± 5.43	37.53 ± 6.39	52.47 ± 2.68	56.97 ± 29.36
Arb Profit ($)	-425 ± 2,836	-3,734 ± 8,684	3,956 ± 7,778	9,945 ± 19,059
Stability	121.78	519.33	182.94	831.77
LP Panics	0.0	1.0	0.0	1.0

Table 1: Comparative Performance Metrics Across AMM Models (mean ± std, n=10)

5.4 Robustness Analysis: Seed Variability

To validate the statistical reliability of our findings, Figure 3 displays the distribution of results across all 10 independent runs (each with different random seeds).

Statistical validation:

- **Uniswap (blue):** Tight clustering in Price Gap distribution (low variance) confirms consistent behavior across random conditions
- **Constant Sum (red):** High variance in all metrics reflects its inherent instability. Even small random variations cause large outcome swings
- **Curve (green):** Moderate spread, with some outliers in arbitrage profit (likely due to specific random market sequences)
- **Arbitrage Profit:** Wide spread in High Volatility scenario (visible in right panel) indicates path-dependent outcomes. Some seeds generate favorable arbitrage sequences, others do not

The boxplots demonstrate that our aggregated results (mean ± std) are not artifacts of a single lucky/unlucky run, but represent genuine performance characteristics reproducible across diverse market conditions.

5.5 Cross-AMM Comparison

Our results reveal stark performance differences between AMM designs:

5.5.1 Constant Sum - Catastrophic Failure

The CSMM model exhibited complete breakdown for volatile ETH/USDC pairs:

- **Baseline price gap:** 5,295 USDC (1,014× worse than Uniswap)

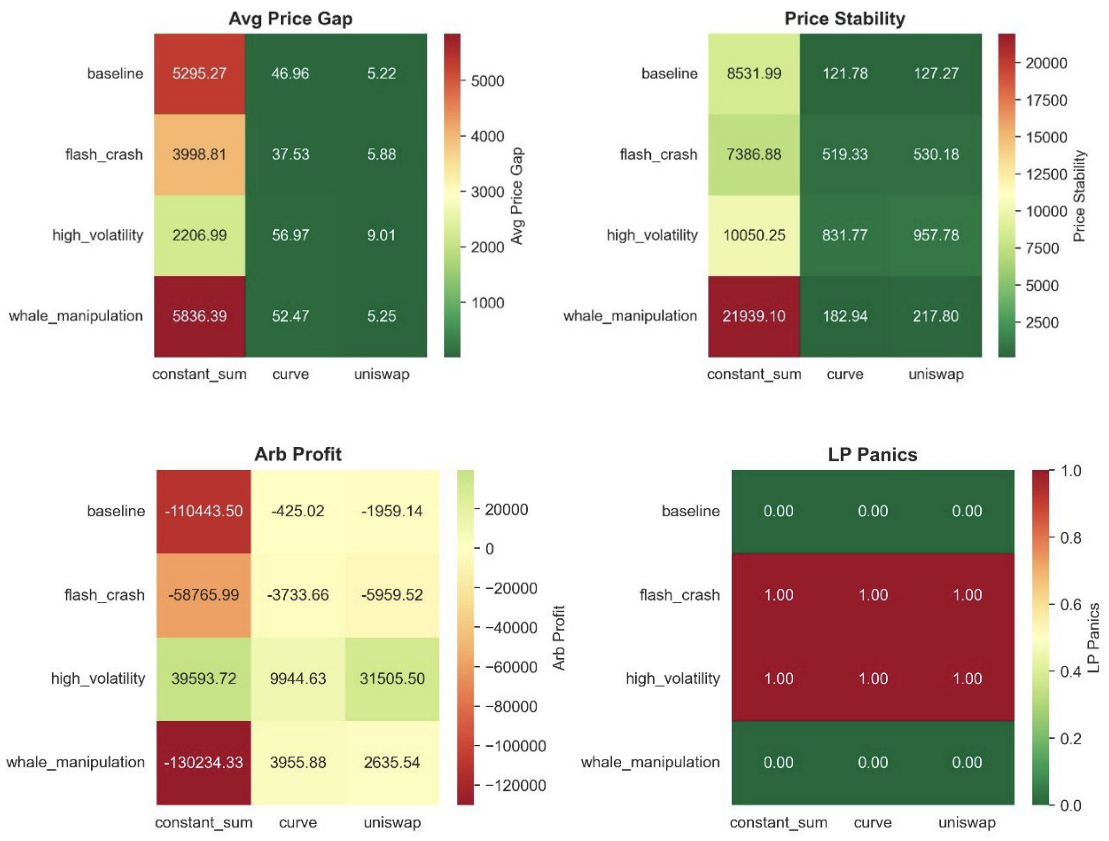

Figure 2: Performance Heatmaps Across AMM Models and Scenarios. Green indicates favorable performance, red indicates poor performance. Note the extreme values in Constant Sum columns, highlighting its unsuitability for volatile pairs.

- **Arbitrageur losses:** -\$110,444 (unable to profit from inefficiency)
- **Instability:** Standard deviation of 8,532 (67× Uniswap's baseline)

This confirms theoretical predictions: CSMM is **unsuitable for non-pegged assets**.

5.5.2 Curve - Moderate Efficiency

Curve's hybrid approach showed intermediate performance:

- **Price gap:** 46.96 USDC (9× worse than Uniswap, but 113× better than CSMM)
- **Comparable stability:** 121.78 (similar to Uniswap's 127.27)
- **Design mismatch:** Curve is optimized for stablecoins (USDC/USDT), not ETH/USDC

5.5.3 Uniswap v2 - Best for Volatile Pairs

The CPMM formula proved most robust for correlated assets with price volatility:

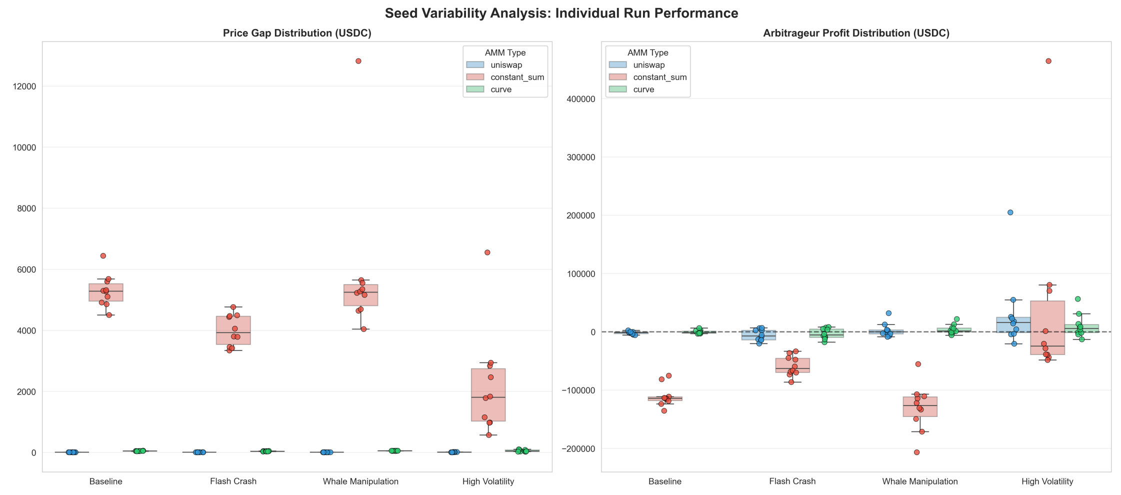

Figure 3: Distribution of Performance Metrics Across 10 Independent Runs. Boxplots show quartiles, while individual points represent single run outcomes. Tight clustering indicates robust, reproducible results.

- Lowest price gaps across all scenarios
- Consistent arbitrage opportunities (positive profit in high volatility)
- Self-correcting mechanism via hyperbolic curve

6 Conclusions

6.1 Key Findings

Our stress-testing simulation revealed both strengths and limitations across all three AMM designs. For Uniswap's Constant Product Market Maker specifically:

Strengths:

- **Self-correcting:** Arbitrageurs effectively close price gaps, even after 50% crashes
- **Resilient k:** Pool invariant grows steadily via fees, maintaining long-term solvency
- **Predictable slippage:** CPMM formula ensures traders cannot completely drain reserves

Limitations:

- **Whale vulnerability:** Price stability degraded 71% under manipulation attacks
- **Extreme volatility:** Price gap increased 73% in sustained chaos, with stability degrading 7.5× compared to baseline
- **Impermanent loss:** LPs suffer losses during volatile periods

6.2 Implications for DeFi

This work demonstrates that while AMMs provide decentralized liquidity, they remain vulnerable to coordinated liquidity withdrawals, market manipulation by well-capitalized actors, and sustained high volatility. Our experiments reveal that these threats are not theoretical: Flash Crash scenarios triggered panic withdrawals across all AMM types, Whale Manipulation degraded price stability, and High Volatility conditions amplified price gaps. These findings underscore the fundamental tension between algorithmic efficiency and systemic resilience in decentralized finance.