Discrete Automated Market Maker (DAMM): A Concentrated Liquidity Protocol

mhatal[1] and fiftyeightandeight[1]

[1]ALEX Team

March 4, 2025

Abstract

This whitepaper presents the Discrete Automated Market Maker (DAMM), an enhanced implementation of Automated Market Making optimized for concentrated liquidity. The system employs an invariant function with virtual liquidity parameters that enable the concentration of liquidity within specified price intervals. This approach significantly improves capital efficiency compared to traditional AMMs while maintaining robust mathematical properties.

1 Introduction

The Discrete Automated Market Maker (DAMM) contract represents an enhanced implementation of Automated Market Making optimized for concentrated liquidity. The system employs the invariant function $(Vx + x) \cdot (Vy + y) = K$, where appropriately selected virtual liquidity parameters Vx and Vy enable the concentration of liquidity within a specified price interval $[P_{\text{start}}, P_{\text{end}}]$.

2 System Overview

The DAMM contract implements a discrete concentrated liquidity automated market maker with the following features:

2.1 Bin-Based Liquidity Concentration

- Liquidity is organized into discrete price bins/ticks
- Supports multiple bin sizes (1%, 5%, 10%, 20%)
- Allows liquidity providers to concentrate their capital in specific price ranges

2.2 Pool Management

- Create trading pools between any two tokens
- Configure fee parameters (fee rates for both tokens and rebate rates)
- Administrative controls to pause/sunset pools when needed

2.3 Position Management

- Add liquidity to specific price ticks
- Add to multiple positions in a single transaction
- Reduce positions by percentage
- Support for batch operations to manage multiple positions efficiently

2.4 Swapping Mechanisms

- Immediate-or-cancel (IOC) swaps
- Fill-or-kill (FOK) swaps
- Cross-tick swaps that route through multiple price bins
- Support for partial fills
- Configurable price limits for slippage protection

2.5 Fee Structure

- Configurable fee rates for each token
- Fee rebates to incentivize liquidity providers
- Fee collection mechanism for protocol revenue
- Asymmetric fees to handle different token characteristics

2.6 Virtual Balances

- Uses virtual reserves to maintain price curves within bins
- Implements constant product formula (x*y=k) with virtual balances
- Optimized mathematical formulations for gas efficiency

3 Mathematical Formulation

We define the fundamental invariant function as:

$$(Vx + x) \cdot (Vy + y) = K \tag{1}$$

where Vx and Vy are virtual balance parameters that constrain trading to the price interval $[P_{\text{start}}, P_{\text{end}}]$, with x and y representing the actual token balances in the pool.

$$Vx = f_x(P_{\text{start}}, P_{\text{end}}, x, y) \tag{2}$$

$$Vy = f_y(P_{\text{start}}, P_{\text{end}}, x, y) \tag{3}$$

Without loss of generality, we partition the price domain $(0, \infty)$ into discrete intervals by introducing an integer parameter $tick \in \mathbb{Z}$ and bin size $bs \in \{1, 5, 10, 20\}$. For each $tick$, we define the price interval boundaries as $P_{\text{start}} = (1 + 0.01 \cdot bs)^{tick}$ and $P_{\text{end}} = (1 + 0.01 \cdot bs)^{tick+1}$.

Let $t = \sqrt{1 + 0.01 \cdot bs}$ and $p = P_{\text{start}}$. It follows that:

$$P_{\text{end}} = t^2 \cdot P_{\text{start}} = pt^2 \tag{4}$$

The price of the pair reaches its lower bound P_{start} when token x is fully depleted (i.e., $\Delta x = -x$), and reaches its upper bound P_{end} when token y is fully depleted (i.e., $\Delta y = -y$). Therefore:

$$p = \frac{Vx}{Vy + y + \Delta y} = \frac{Vx^2}{Vx \cdot (Vy + y + \Delta y)} = \frac{Vx^2}{K} \tag{5}$$

$$pt^2 = \frac{Vx + x + \Delta x}{Vy} = \frac{Vy \cdot (Vx + x + \Delta x)}{Vy^2} = \frac{K}{Vy^2} \tag{6}$$

From these equations, we derive:

$$p^2 t^2 = \frac{Vx^2}{Vy^2} \tag{7}$$

$$Vx = p \cdot t \cdot Vy \tag{8}$$

Substituting this relation:

$$pt^2 = \frac{K}{Vy^2} = \frac{(Vx + x)(Vy + y)}{Vy^2} = \frac{(p \cdot t \cdot Vy + x)(Vy + y)}{Vy^2} \tag{9}$$

$$p(t^2 - t)Vy^2 - (x + pty)Vy - xy = 0 \tag{10}$$

Solving this quadratic equation for Vy yields:

$$Vy = \frac{x + pty + \sqrt{(x + pty)^2 + 4p(t^2 - t)xy}}{2p(t^2 - t)} \tag{11}$$

Substituting back to find Vx:

$$Vx = \frac{x + pty + \sqrt{(x + pty)^2 + 4p(t^2 - t)xy}}{2(t - 1)} \tag{12}$$

4 Swap Mechanics

4.1 Swapping x for y Until Price Reaches P_{max}

This is similar to Immediate or Cancel (IOC) order in orderbook, given Δx, swap as much as possible until the price hit P_{max}, where P_{max} is within the price range $[P_{\text{start}}, P_{\text{end}}]$.

$$P_{\text{max}} = \frac{Vx + x + \Delta x_{\text{max}}}{Vy + y - \Delta y} \tag{13}$$

$$K \cdot P_{\text{max}} = (Vx + x + \Delta x_{\text{max}})^2 \tag{14}$$

$$\Delta x_{\text{max}} = \sqrt{K \cdot P_{\text{max}}} - (Vx + x) \tag{15}$$

$$\Delta x_{\text{swappable}} = \min(\max(0, \Delta x_{\text{max}}), \Delta x) \tag{16}$$

$$\Delta y = (Vy + y) - \frac{K}{Vx + x + \Delta x_{\text{swappable}}} \tag{17}$$

4.2 Swapping y for x Until Price Reaches $P_{\mathbf{min}}$

Similarly, given an input quantity Δy, the system executes the swap until the price reaches P_{min}, where P_{min} lies within the interval $[P_{\mathrm{start}}, P_{\mathrm{end}}]$.

$$P_{\mathrm{min}} = \frac{Vx + x - \Delta x}{Vy + y + \Delta y_{\mathrm{max}}} \tag{18}$$

$$\frac{K}{P_{\mathrm{min}}} = (Vy + y + \Delta y_{\mathrm{max}})^2 \tag{19}$$

$$\Delta y_{\mathrm{max}} = \sqrt{\frac{K}{P_{\mathrm{min}}}} - (Vy + y) \tag{20}$$

$$\Delta y_{\mathrm{swappable}} = \min(\max(0, \Delta y_{\mathrm{max}}), \Delta y) \tag{21}$$

$$\Delta x = (Vx + x) - \frac{K}{Vy + y + \Delta y_{\mathrm{swappable}}} \tag{22}$$

5 Liquidity Management

5.1 Adding Liquidity

After adding Δx and Δy to the pool, both the pool size (i.e., liquidity token balance) and token pair price will be affected. The price after liquidity addition is given by:

$$\text{Price} = \frac{V_x' + x + \Delta x}{V_y' + y + \Delta y} \tag{23}$$

The price should be verified to prevent unexpected deviation from external markets. The pool size increases proportionally to the virtual balances:

$$L_p' = L_p \cdot \frac{V_x'}{V_x} = L_p \cdot \frac{V_y'}{V_y} \tag{24}$$

This relationship is derived from the fact that swapping does not change V_x, V_y, or the pool size. When token y is depleted (i.e., $y = 0$), the balance of token x represents the total value of the pool:

$$V_x = \frac{x + pty + \sqrt{(x + pty)^2 + 4p(t^2 - t)xy}}{2(t-1)} = \frac{x}{t-1} \text{ when } y = 0 \tag{25}$$

$$x = (t-1) \cdot V_x \tag{26}$$

$$L_p' = L_p \cdot \frac{x'}{x} = L_p \cdot \frac{(t-1) \cdot V_x'}{(t-1) \cdot V_x} = L_p \cdot \frac{V_x'}{V_x} \tag{27}$$

Similarly, when token x is depleted (i.e., $x = 0$), the balance of token y represents the total value of the pool:

$$V_y = \frac{x + pty + \sqrt{(x + pty)^2 + 4p(t^2 - t)xy}}{2p(t^2 - t)} = \frac{y}{t-1} \text{ when } x = 0 \tag{28}$$

$$y = (t-1) \cdot V_y \tag{29}$$

$$L_p' = L_p \cdot \frac{y'}{y} = L_p \cdot \frac{(t-1) \cdot V_y'}{(t-1) \cdot V_y} = L_p \cdot \frac{V_y'}{V_y} \tag{30}$$

5.2 Reducing Liquidity

DAMM provides flexible options for reducing liquidity positions:

- **Percentage-Based Reduction**: Positions can be reduced by a specified percentage rather than absolute amounts
- **Batch Operations**: Multiple positions can be reduced in a single transaction
- **Proportional Withdrawal**: When reducing a position, tokens are withdrawn proportionally to maintain the price ratio

When a liquidity provider reduces their position by a percentage p, the following occurs:

$$\Delta x = p \cdot x \tag{31}$$

$$\Delta y = p \cdot y \tag{32}$$

This proportional reduction ensures that the price ratio of the trading pair remains unchanged, maintaining the integrity of the price curve within the bin.

6 Comparison with Other Concentrated Liquidity Protocols

DAMM offers several advantages when compared to other concentrated liquidity automated market makers. Table 1 below provides a comparison of key features across major protocols.

6.1 Key Advantages of DAMM

- **Simplified Position Management**: More straightforward than NFT-based positions while maintaining concentrated liquidity benefits
- **Enhanced Price Protection**: Superior protection against manipulation during pool initialization
- **Flexible Fee Structure**: Asymmetric fees to better handle different token characteristics
- **Predictable Liquidity Distribution**: Easier to reason about for both traders and liquidity providers
- **Batch Operations**: Efficient management of multiple positions in a single transaction
- **Multiple Swap Types**: Support for both IOC and FOK swap types with partial fills

7 License

This project is licensed under the Business Source License 1.1 (BUSL-1.1).

The Business Source License is a non-open source license that allows free usage for non-production purposes but restricts commercial use for a specified period, after which it converts to a specified open source license.

Feature	DAMM	Uniswap V3	Meteora DLMM	Trader Joe v2
Liquidity Concentration	Discrete price bins (1%, 5%, 10%, 20%)	Continuous price curves	Dynamic liquidity allocation	Discrete bins (Liquidity Book)
Position Representation	Fungible liquidity tokens	Non-fungible positions (NFTs)	Fungible tokens	Bin-specific tokens
Price Protection	Virtual balance mechanism	No built-in protection	Limited information	Limited information
Capital Efficiency	High with simpler management	Very high (up to 4000x)	High with dynamic fees	High with bin-based approach
Fee Structure	Configurable per-token fees with rebates	Three fixed fee tiers	Dynamic volatility-based fees	Variable bin-based fees
Position Management	Batch operations supported	Individual management	Limited information	Bin-specific management
Cross-Range Swaps	Efficient multi-bin routing	Requires path optimization	Limited information	Bin traversal
Swap Types	Both IOC and FOK supported	Only standard swaps	Limited information	Limited information
Implementation	Clarity language on Stacks	Solidity on Ethereum	Rust on Solana	Solidity on Avalanche

Table 1: Comparison of DAMM with other concentrated liquidity protocols