ALEX AMM V3: Concentrated Liquidity Automated Market Maker

mhatal[1] and fiftyeightandeight[1]

[1]ALEX Team

February 27, 2025

Abstract

This whitepaper presents the ALEX AMM V3, an enhanced implementation of Automated Market Making (AMM) 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 ALEX AMM V3 contract represents an enhanced implementation of Automated Market Making (AMM) 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 AMM Pool V3 contract implements a 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
- 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

2.5 Fee Structure

- Configurable fee rates for each token
- Fee rebates to incentivize liquidity providers
- Fee collection mechanism for protocol revenue

2.6 Virtual Balances

- Uses virtual reserves to maintain price curves within bins
- Implements constant product formula (x*y=k) with virtual balances

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 $[P_{\text{start}}, P_{\text{end}}]$.

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

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

Without loss of generality, we partition the price domain $(0, \infty)$ by introducing an integer parameter *tick* and bin size *bs*. For each *tick*, we define the price interval 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 equals P_{start} when balance x is fully depleted (i.e., $\Delta x = -x$), and equals P_{end} when balance 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:

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

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

Substituting this relation:

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

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

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

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

4 Swap Mechanics

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

This mechanism resembles an Immediate-or-Cancel (IOC) order in traditional order books. Given an input quantity Δx, the system executes the swap until the price reaches P_{max}, where P_{max} lies within the interval $[P_{\text{start}}, P_{\text{end}}]$.

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

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

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

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

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

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 $[P_{\text{start}}, P_{\text{end}}]$.

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

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

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

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

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

5 Liquidity Management

5.1 Adding Liquidity

When we scale the existing pool balances proportionally by a factor σ, the invariant function becomes:

$$(\sigma Vx + \sigma x) \cdot (\sigma Vy + \sigma y) = \sigma^2 K \tag{23}$$

$$\text{Price} = \frac{\sigma Vx + \sigma x}{\sigma Vy + \sigma y} = \frac{Vx + x}{Vy + y} \tag{24}$$

This demonstrates that proportional changes to pool balances preserve the price ratio. Consequently, when adjusting liquidity for a given Δx, we have:

$$\Delta y = \frac{y}{x} \cdot \Delta x \tag{25}$$

5.2 Initializing a New Pool

When initializing a new pool, we implement a price-protective mechanism:

1. We calculate a target price as the geometric mean of the minimum and maximum acceptable prices:

$$P_{\text{target}} = \sqrt{P_{\text{min}} \cdot P_{\text{max}}} \tag{26}$$

2. We verify that this target price falls within the bin's price range:

$$P_{\text{start}} \leq P_{\text{target}} \leq P_{\text{end}} \tag{27}$$

3. We balance the provided token amounts according to the target price:

$$\text{bal}_x = \min\left(x_{\text{max}}, y_{\text{max}} \cdot P_{\text{target}}\right) \tag{28}$$

$$\text{bal}_y = \min\left(y_{\text{max}}, \frac{x_{\text{max}}}{P_{\text{target}}}\right) \tag{29}$$

4. The total supply of liquidity tokens is calculated as the geometric mean of the balanced amounts:

$$\text{total\_supply} = \sqrt{\text{bal}_x \cdot \text{bal}_y} \tag{30}$$

This approach ensures that:

- The initial price is fair and within the acceptable range
- The token amounts are balanced according to the target price
- The pool is initialized with sufficient liquidity
- Price manipulation is prevented by using the geometric mean of price bounds

5.3 Reducing Liquidity

Analogous to liquidity addition, proportional reduction of liquidity maintains the price ratio of the trading pair.

6 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.