CBMM Pool: A Constant Burn Market Maker

@mmatdev
Access Protocol

November 26, 2025

Abstract

This whitepaper describes the Constant Burn Market Maker (CBMM) pool, an extension of the standard constant product market maker (CPMM) that uses a virtual quote reserve and built-in burns to enable translating verifiable off-chain activity into on-chain base token supply reduction and price impact. We outline the core mathematical model and safety conditions, and describe a Solana implementation with Continuous Conditional Buybacks (CCB) and practical controls such as burn caps, throttling, and authorization.

1 Introduction

Constant function market makers (CFMMs) have become a foundational primitive for decentralized exchanges [1]. Variants include constant product market makers (CPMMs) such as Uniswap v1 [2], concentrated liquidity market makers (CLMMs) like Uniswap v3 [3], and dynamic automated market makers (DAMMs) [4]. Bonding curves [5], popularized by platforms like Pump.fun, represent another approach where token price is determined by a deterministic curve based on supply.

Existing mechanisms provide no built-in way to translate off-chain activity into on-chain price impact. The only mechanism available to drive positive price action is to coordinate manual buy-and-burn operations, where participants purchase tokens and burn them to reduce supply. This requires coordination, creates friction, and does not automatically link rewards to verifiable off-chain behavior. However, naively bolting burns onto CFMMs with real reserves creates value-extraction vectors: an attacker can execute sequences of operations that siphon collateral from the pool unless this functionality is carefully constrained.

We present the Constant Burn Market Maker (CBMM), a mechanism that enables tying off-chain events via on-chain burns to underlying asset supply reduction. CBMM utilizes a virtual quote token reserve that is used to support controlled burns without violating pool invariant constraints and enables creating one-sided launch pools. These burns directly increase price proportionally to the amount of base tokens held outside the pool and can be tied to off-chain events. Moreover, to compensate for the virtual reserve reduction that accompanies burns, we implement a real queue token reserve top-up mechanism called Continuous Conditional Buybacks (CCB), which route a portion of trading fees into the real token reserve, effectively increasing collateralization and increasing price further. On top of this, we introduce burn caps, symmetric quote-token fees, and pacing and authorization controls that jointly enforce formal safety guarantees against these extraction loops.

The remainder of this paper is structured as follows. Section 2 develops the mathematical model, derives trading and burning formulas, and analyzes price impact and mitigation of possible attacks. Section 3 describes the on-chain program implementation considerations, including safety mechanisms and configuration considerations.

2 Mathematical Model

In this section we will describe the mathematical model of the CBMM pool. Let us first define the key terms used throughout this section. Let there be a CPMM pool with a real quote token reserve A (backed by actual assets), real base token reserve B and virtual quote token reserve V (not backed by assets, used to set the initial price and liquidity). The pool trading mechanics are defined by the constant product invariant $k = (A + V)B$. We say that this pool is a **CBMM pool** if it implements the base token burn functionality tied to the virtual reserve reduction as described in subsection 2.2.

The CBMM pool is said to be **insolvent** if its quote token reserves are insufficient to accommodate the sale of all outstanding base tokens; otherwise it is said to be **solvent**. The goal is to always keep the CBMM pool solvent which can be achieved by adjusting the virtual reserve after each burn.

In this whitepaper the CBMM is designed as a base token rollout mechanism, so we do not need to account for any existing supply outside the pool. Moreover, we assume that the initial real quote token reserve A is zero and, therefore, the virtual quote token reserve V sets the initial price of the base token.

2.1 Trading

Buys and sells in CBMM follow the mechanics of the standard CPMM with a quote token virtual reserve; we include them here for completeness. The pre-trade (buy or sell) values are

$$A = A_0, \qquad B = B_0, \qquad V = V, \qquad k = (A_0 + V)B_0.$$

During trading, the invariant k and the virtual reserve V do not change. The amount of base tokens b received by the user when spending ΔA quote tokens follows from

$$k = (A_0 + \Delta A + V)(B_0 - b), \tag{1}$$

which gives:

$$b = B_0 - \frac{k}{A_0 + \Delta A + V}. \tag{2}$$

Similarly, the amount of quote tokens a received by the user when spending ΔB base tokens follows from

$$k = (A_0 - a + V)(B_0 + \Delta B), \tag{3}$$

which gives:

$$a = A_0 + V - \frac{k}{B_0 + \Delta B}. \tag{4}$$

The price of base tokens P (quote tokens per base token) is derived from the marginal rate of exchange. Starting with the invariant $k = (A + V)B$ and holding k constant, we differentiate with respect to B:

$$\frac{d}{dB}[(A + V)B] = \frac{dA}{dB} \cdot B + (A + V) = 0, \tag{5}$$

which yields $\frac{dA}{dB} = -(A + V)/B$. The price is the negative of this derivative:

$$P = -\frac{dA}{dB} = \frac{A + V}{B}. \tag{6}$$

2.2 Base Token Reserve Burning

This section describes the necessary conditions for the CBMM pool to remain solvent after a base token reserve reduction.

Let the initial state of the pool be

$$A_0 = 0, \qquad B_0 = T, \qquad V_0 = V, \qquad k_0 = (0 + V)T = VT.$$

Assume, without loss of generality, that trading occurs before the burn, lowering the base token reserve by $x \geq 0$. The post-trade values are

$$A_1 = \frac{Vx}{T - x}, \qquad B_1 = T - x, \qquad V_1 = V, \qquad k_1 = VT.$$

Now, if we burn y base tokens with $0 \leq y < T - x$, the post-burn state is

$$A_2 = \frac{Vx}{T - x}, \qquad B_2 = T - x - y, \qquad V_2 \text{ to be determined}, \qquad k_2 = (A_2 + V_2)B_2.$$

Let's find the condition for the virtual reserve V_2 to ensure the pool is solvent. This means that if everyone sells their base tokens back to the pool, there must be sufficient quote tokens to satisfy the sale. The post-sale state must satisfy

$$A_3 \geq 0, \qquad B_3 = T - y, \qquad V_3 = V_2, \qquad k_3 = (A_3 + V_3)B_3.$$

From $k_2 = k_3$ and $k_3 \geq V_3 B_3$, we obtain a bound on V_2 that ensures the pool is solvent:

$$V_2 \leq \frac{V(B_1 - y)}{B_1} = \frac{V(T - x - y)}{T - x}. \tag{7}$$

Thus, after every burn, the virtual reserve must be adjusted downward to ensure solvency. A natural choice is to set V_2 to the maximum value satisfying the bound, which minimizes the price impact of the virtual reserve reduction.

2.2.1 Price impact of the burn

Denote the price before burn as $P_1 = \frac{A_1+V_1}{B_1}$ and the price after burn as $P_2 = \frac{A_2+V_2}{B_2}$. Substituting the values from the previous section, where $B_1 = T - x$ and $V_2 = \frac{V(T-x-y)}{T-x}$, we obtain

$$P_1 = \frac{A_1 + V_1}{B_1} = \frac{\frac{Vx}{T-x} + V}{T - x} = \frac{VT}{(T - x)^2},$$

$$P_2 = \frac{A_2 + V_2}{B_2} = \frac{\frac{Vx}{T-x} + \frac{V(T-x-y)}{T-x}}{T - x - y} = \frac{V(T - y)}{(T - x)(T - x - y)}.$$

The relative price impact of the burn is then:

$$\frac{P_2 - P_1}{P_1} = \frac{xy}{T(T - x - y)}. \tag{8}$$

This formula shows that the relative price impact is proportional to the product xy of base tokens held outside the pool x and base tokens burned y, divided by $T(T - x - y)$. The impact increases with x, meaning burns have greater price impact when more base tokens have been purchased. If $x = 0$ (no base tokens purchased), the burn has no price impact, as expected.

The solvency adjustment $V_2 \leq V(T - x - y)/(T - x)$ ensures that $V_2 < V$ for any $y > 0$. This reduction in the virtual reserve can affect the worst-case exit price. The worst-case exit price refers to the marginal price when all outstanding base tokens (post-burn) are sold back to the pool, driving the real quote token reserve A toward zero. At this limit, the price approaches $V_2/(T-y)$. When $x > 0$ and $y > 0$, this worst-case exit price can be below the initial anchor price V/T set by the starting virtual reserve V. However, when $x = 0$ (no base tokens purchased before the burn), the worst-case exit price equals the initial price, as $V_2 = V(T-y)/T$ and $V_2/(T-y) = V/T$.

Operationally, burns reduce the base token reserve B, tightening depth and available liquidity at the new state.

2.3 Token Reserve Top-up

Burns reduce the base token reserve B and force a downward adjustment of the virtual reserve V. This causes a liquidity reduction and price raise offset by the price curve getting steeper and the initial price being reduced. This unwanted side effect can be undone by adding quote tokens to the real quote token reserve A which we call **top-up**. This subsection derives the exact top-up amount needed to restore the base token starting price without changing the logic introduced earlier.

Let T_i denote the total base token supply after i operations. The pre-trade state is

$$A_0 = 0, \qquad B_0 = T_0, \qquad V_0 = V, \qquad T_0 = T, \qquad k_0 = (A_0 + V_0)B_0 = V_0 T_0.$$

After trades that extract x base tokens from the pool, the state is

$$A_1 = A_0 + \frac{(A_0 + V_0)x}{B_0 - x} = \frac{Vx}{T-x}, \qquad B_1 = B_0 - x, \qquad V_1 = V_0, \qquad T_1 = T_0, \qquad k_1 = k_0.$$

Next, burn y base tokens ($0 \le y < B_1$). Let V_2 denote the virtual reserve enforced by the solvency condition in Section 2.2. The post-burn state is

$$A_2 = A_1, \qquad B_2 = B_1 - y, \qquad V_2 \le V_1, \qquad T_2 = T_1 - y, \qquad k_2 = (A_2 + V_2)B_2.$$

Our target is to find a virtual reserve V_{opt} and a real quote token amount A_{opt} that keep the worst-case exit price – the price when all outstanding base tokens are sold back – at the original value $P_0 = V_0/T_0$. Let's denote the needed top-up amount to achieve the optimal state as M. If we manage to achieve this top-up, the pool state will be

$$A_3 = A_2 + M = A_{\text{opt}}, \quad B_3 = B_2, \quad V_3 = V_{\text{opt}}, \quad T_3 = T_2, \quad k_3 = k_{\text{opt}} = (A_{\text{opt}} + V_{\text{opt}})B_2.$$

If everyone exits after the top-up, the pool holds $B_3 = T_2$ base tokens and zero real quote tokens, so the price becomes $P_3 = V_{\text{opt}}/T_2$. Enforcing $P_3 = P_0$ yields

$$\frac{V_0}{T_0} = \frac{V_{\text{opt}}}{T_2} \implies V_{\text{opt}} = \frac{T_2}{T_0}V_0 = \frac{T-y}{T}V_0. \tag{9}$$

With this V_{opt} value, the invariant that corresponds to the desired price profile is

$$k_{\text{opt}} = (0 + V_{\text{opt}})T_2 = V_{\text{opt}}T_2 = (A_{\text{opt}} + V_{\text{opt}})B_2. \tag{10}$$

This implies the target real quote token reserve

$$A_{\text{opt}} = \frac{k_{\text{opt}}}{B_2} - V_{\text{opt}} = \frac{T_1 - y}{T_1}V_1\frac{T_1 - B_1}{B_1 - y}. \tag{11}$$

The second equality follows by substituting $k_{\text{opt}} = V_{\text{opt}}T_2$, $B_2 = B_1 - y$, $T_2 = T_1 - y$, $T_1 = T$, and $T_1 - B_1 = x$ into the first expression. The required quote token top-up amount M is

$$M = A_{\text{opt}} - A_2 = \frac{T-y}{T}V\frac{T-(B_0-x)}{(B_0-x)-y} - \left(A_0 + \frac{(A_0+V)x}{B_0-x}\right). \tag{12}$$

Or in other terms for simple on-chain calculation:

$$M = \frac{A_1^2}{(A_1 + V_1)(B_1 - y)}. \tag{13}$$

2.3.1 Trading impact on required top-up amount

If the top-up does not happen atomically with the burn and is delayed, some trading may occur in the meantime. This impacts the required top-up amount. Suppose that after the burn, instead of applying the top-up immediately, trades shift the base token reserve from B_2 to $B' = B_2 + \Delta B$ with $-B_2 < \Delta B \leq T_2 - B_2$. Positive ΔB corresponds to net sells back into the pool, while negative ΔB captures net buys.

The target optimal reserve A'_{opt} that preserves the desired price profile at the new base token reserve B' is tied to k_{opt} and V_{opt} which remain unchanged by trading:

$$A'_{\text{opt}} = \frac{k_{\text{opt}}}{B'} - V_{\text{opt}}. \tag{14}$$

Meanwhile, the actual quote token reserve A'_{real} at base token reserve B' induced by the current invariant $k_2 = (A_2 + V_2)B_2$ becomes

$$A'_{\text{real}} = \frac{k_2}{B'} - V_2. \tag{15}$$

The updated top-up requirement after the trades is therefore

$$M' = A'_{\text{opt}} - A'_{\text{real}} = \frac{k_{\text{opt}} - k_2}{B'} + (V_2 - V_{\text{opt}}). \tag{16}$$

Buys ($\Delta B < 0$) shrink B', amplifying the first term and increasing M'; sells ($\Delta B > 0$) expand B' and dampen the same term, but the additive offset $(V_2 - V_{\text{opt}}) < 0$ keeps the gap positive. At the boundary case $\Delta B = T_2 - B_2$ the expression is undefined. In that state no base tokens are left outside the pool, so the pool price can be without any unwanted side effect reset to the original initial price by setting $V = V_{\text{opt}}$ without any real quote token top-up.

2.3.2 Partial Top-ups

Suppose only $M' < M$ quote tokens are available for the top-up immediately after the burn. Injecting M' raises the real quote token reserve to $A_{\text{new}} = A_2 + M'$ while the base token reserve stays at B_2. The invariant is therefore

$$k_{\text{new}} = (A_{\text{new}} + V_{\text{new}})B_2, \tag{17}$$

where V_{new} is the virtual reserve after the partial top-up and is unknown. We do not target a specific k_{new}; instead we push the post-injection price as high as pool solvency requirement allows. When every outstanding base token ($T_2 - B_2$ in total) is sold back, the goal is for the pool to reach $A = 0$. This corresponds to the invariant

$$k_{\text{new}} = V_{\text{new}} T_2. \tag{18}$$

Equating both expressions for k_{new} yields the reserve closest to V_{opt} under a partial top-up that keeps the pool solvent:

$$V_{\text{new}} = \frac{A_{\text{new}} B_2}{T_2 - B_2}. \tag{19}$$

When $T_2 > B_2$, this choice keeps the pool solvent and maximizes the achievable price lift given the available collateral. Any subsequent injection simply repeats the calculation with updated A_{new} and moves V_{new} closer to V_{opt} until the full top-up is completed.

As already discussed in Section 2.3.1, if trades drive $T_2 = B_2$ (no base tokens outside the pool), V_{new} can be reset directly to V_{opt} because no outstanding holders remain.

2.4 Value Extraction Considerations and Mitigation Strategies

A critical concern for CBMM pools is that adversaries may attempt to extract value from the real quote token reserve A through strategic manipulation of burns. An attacker can execute a buy–burn–sell loop: purchase base tokens, trigger a burn (which reduces the virtual reserve V), and then sell the base tokens back, potentially realizing a net profit at the expense of the pool's real collateral. This section quantifies the exact profit that an adversary can realize through such attacks and outlines mitigation strategies. We first analyze the basic attack without fees, then show how symmetric quote-token fees can suppress the attack, and finally demonstrate that when top-ups are involved, additional constraints such as per-burn caps are necessary to prevent profitable exploitation.

2.4.1 Attack model and payoff derivation

Consider a pool in state (A, B, V) with invariant $k = (A + V)B$ and some base tokens in circulation. Total base token supply is $T > B$. An adversary executes the following steps:

1. **Buy x base tokens.** The adversary buys x base tokens paying A_{in} quote tokens. According to the standard CBMM trading logic

$$A_{\text{in}} = (A + V)\frac{x}{B - x}, \qquad 0 < x < B. \tag{20}$$

   The post-buy reserves are $A_1 = A + A_{\text{in}}$ and $B_1 = B - x$.

2. **Trigger a burn of y base tokens** from the pool (with $0 < y < B - x$), enforcing solvency by reducing the virtual reserve to

$$V_2 = \frac{V(B - x - y)}{B - x}, \tag{21}$$

   so that the new invariant remains $k_2 = (A_1 + V_2)(B - x - y)$.

3. **Sell the x base tokens back.** Using the sell formula from Section 2, the attacker receives

$$A_{\text{out}} = \frac{(A_1 + V_2)x}{B - y} \tag{22}$$

   quote tokens while the pool returns to base token reserve $B - y$.

Combining Equations (20) to (22) yields a closed-form expression for the net quote token profit:

$$\Pi(x, y) = A_{\text{out}} - A_{\text{in}} = \frac{Axy}{(B - x)(B - y)}. \tag{23}$$

The profit formula Equation (23) reveals several properties of the attack. Most critically, whenever $A > 0$, the profit $\Pi(x, y) \geq 0$ for all admissible x and y, meaning the attack is always profitable (or at least break-even) whenever there is any real reserve. This is the fundamental vulnerability: any real quote token reserve can be extracted through a buy–burn–sell loop. The extractable value is zero only when $A = 0$, meaning no real reserve implies no profit opportunity. Profit grows jointly with both x and y, reaching its maximum as both approach B. In the limit $\lim_{x,y \to B} \Pi(x, y) = A$, an adversary buying and burning nearly all tokens from the pool reserve can extract almost the entire real reserve. The attack strictly consumes existing collateral: after selling back, the pool retains only $A - \Pi$ quote tokens. This implies that any mechanism increasing A must implement a compensation logic to prevent the adversary from profiting.

2.4.2 Attack model with symmetric base-token fees

To actively suppress the buy–burn–sell loop, we impose an $N\%$ fee on every trade, collected entirely in the quote token. This fee penalizes extraction attempts in the following way. Define the fee multiplier

$$q = \frac{100}{100 - N} \tag{24}$$

All fees are skimmed from the trader's quote token transfers: buy-side fees are removed before quote tokens enter the pool, and sell-side fees are shaved off before proceeds hit the attacker's wallet.

Step 1: buy x base tokens. To withdraw x base tokens $(0 < x < B)$, the trader must send S_{buy} quote tokens so that the post-fee deposit equals the CBMM requirement. Solving

$$(A + V)B = (A + S_{\text{buy}}/q + V)(B - x) \tag{25}$$

gives

$$S_{\text{buy}} = q\,\frac{(A + V)x}{B - x}. \tag{26}$$

The pool itself still sees $A_1 = A + \frac{(A+V)x}{B-x}$ and $B_1 = B - x$, identical to the no-fee path, while the attacker cost is increased by the fees.

Step 2: burn y base tokens. Burning y base tokens $(0 < y < B_1)$ forces the same solvency adjustment as before:

$$A_2 = A_1, \qquad B_2 = B - x - y, \qquad V_2 = \frac{V(B - x - y)}{B - x}.$$

Step 3: sell the x base tokens back. The on-chain invariant after the burn is

$$k_{\text{burn}} = (A_2 + V_2)B_2. \tag{27}$$

Trading x base tokens back into the pool yields the pre-fee quote token outflow S_{pool} defined by

$$(A_2 + V_2)B_2 = (A_2 - S_{\text{pool}} + V_2)(B_2 + x), \tag{28}$$

which solves to

$$S_{\text{pool}} = \frac{x}{(B - x)(B - y)}\Big[B(A + V) - Vy\Big]. \tag{29}$$

The fee clips a factor $1/q$ from the output, so the attacker actually receives

$$S_{\text{sell}} = \frac{1}{q}\,S_{\text{pool}} = \frac{1}{q}\,\frac{x}{(B - x)(B - y)}\Big[B(A + V) - Vy\Big]. \tag{30}$$

Net payoff and burn threshold. The round-trip profit becomes

$$\Pi_{\text{fee}}(x, y) = S_{\text{buy}} - S_{\text{sell}} = \frac{x}{q(B - x)(B - y)}\Big[q^2(A + V)(B - y) - B(A + V) + Vy\Big]. \tag{31}$$

Fees suppress the attack whenever the bracketed term is non-negative. Solving the linear inequality for y yields the maximum burn size that still keeps the attacker under water:

$$y \le y_{\max} = \frac{(q^2 - 1)B(A + V)}{q^2(A + V) - V}, \tag{32}$$

The main observation is that the profitability condition depends only on the burn size y relative to the pool parameters; the buy size x affects the magnitude of profit but not its sign, so the threshold $y_{\max}$ is independent of the attacker's purchase amount.

Assuming $A > 0$ (otherwise there is nothing to extract) keeps the denominator positive. The bound scales linearly with the base token reserve B; the multiplier depends on the fee rate (via q), the virtual reserve V, and the real quote token reserve A. Setting $N = 0$ (so $q = 1$) collapses the bound to $y_{\max} = 0$, matching the logic in Equation (23). For $q > 1$, the attack is eliminated whenever the configured burn allowance lies below $y_{\max}$.

Even when fees prevent the attacker from profiting (i.e., when $y \leq y_{\max}$), the pool can still be negatively impacted by the attack. The burn reduces the base token reserve and forces a downward adjustment of the virtual reserve, which reduces liquidity and can lower the worst-case exit price. Moreover, the fees collected during the attack may not fully compensate for these negative effects. Therefore, it would be reasonable to consider allowing only smaller burn sizes y than $y_{\max}$ to further limit the pool's exposure to such attacks, even when they are not profitable for the attacker.

2.4.3 Attack model with symmetric base-token fees and topup

We extend the attack model to account for a quote token reserve top-up that occurs after the burn. The top-up increases the real quote token reserve and adjusts the virtual reserve for optimal utilization as described in Section 2.3. We keep the same notation as before: the pool has an initial state with real quote token reserve $A \geq 0$, base token reserve $B > 0$, virtual reserve $V > 0$, total base token supply $T \geq B$, and invariant $k = (A + V)B$. Fees are charged symmetrically on the quote token at rate $N\%$, with fee multiplier

$$q = \frac{100}{100 - N} > 1. \tag{33}$$

An adversary executes the buy–burn–topup–sell loop: (i) buys x base tokens ($0 < x < B$) at cost $S_{\text{buy}} = q(A + V)x/(B - x)$, (ii) triggers a burn of y base tokens ($0 < y < B - x$) followed by a top-up, and (iii) sells the x base tokens back. Since top-ups increase the real quote token reserve, they increase the attacker's profit from selling base tokens back. The worst case for the protocol occurs when the maximal top-up M (as defined in Section 2.3) is applied, as this maximizes the attacker's gain.

Concrete example: profitable attack despite fees. To see that fees alone do not suffice when top-ups are unconstrained, consider the following concrete example. Take

$$A = 100, \quad B = 100, \quad V = 10,$$

so that $A + V = 110$ and the invariant is $k = (A + V)B = 11{,}000$. From $k = VT$ we get a total supply $T = k/V = 1100$. Set a symmetric fee of $N = 5\%$, so $q = 100/95 \approx 1.0526$.

The attacker executes the following steps:

1. **Buy $x = 90$ base tokens.** This costs $S_{\text{buy}} \approx 1042.11$ quote tokens, leaving $B_1 = 10$ base tokens in the pool.

2. **Burn $y = 1$ base token.** The post-burn state has $B_2 = 9$ base tokens in the pool. Note that $y = 1$ satisfies the condition from the previous subsection: with $q^2 = (100/95)^2 \approx 1.108$, we have $y_{\max} = \frac{(q^2-1)B(A+V)}{q^2(A+V)-V} \approx 10.62$ (calculated using the initial $B = 100$). Since $y = 1 < y_{\max} \approx 10.62$ and $y = 1 < B_1 = 10$, the burn is feasible and the attack would be unprofitable without top-ups.

3. **Apply maximal top-up M.** The top-up mechanism from Section 2.3 injects the optimal amount $M \approx 120.01$ to the real quote token reserve to restore the target price profile and adjusts $V = V_{\text{opt}} \approx 9.9909$

4. **Sell the $x = 90$ base tokens back.** The attacker receives $S_{\text{sell}} \approx 1053.64$ quote tokens after fees.

The net profit is approximately $11.53 > 0$ quote tokens. This example satisfies all feasibility constraints, yet leads to a profitable buy–burn–topup–sell loop even with a 5% symmetric fee. The issue is that the top-up injects enough real quote tokens to more than offset the fee friction.

This motivates one additional design constraint besides adding a sufficiently large symmetric fee - per-burn limit on the number of base tokens that may be burned. The derivation of the formula describing the relationship of these two parameters follows.

Lemma 2.1 (Profit formula with top-ups)**.** Let there be a CBMM pool in state (A, B, V) with invariant $k = (A + V)B$, total base token supply $T > B$ and symmetric quote-token fee $N\%$ and fee multiplier $q = 100/(100 - N) > 1$. After an attacker buys x base tokens, triggers a burn of y base tokens, applies a top-up $0 \le N \le M$ (as defined in Section 2.3) and sells the x base tokens back, the maximum possible round-trip profit is

$$\Pi_{\text{topup}}(x, y, N) = \frac{x(T-y)^2 V}{qT(B-y)(B-x-y)} - q\,\frac{(A+V)x}{B-x}. \tag{34}$$

and happens when $N = M$.

Proof. After the attacker buys x base tokens, the pool state is

$$A_1 = A + \frac{(A+V)x}{B-x}, \qquad B_1 = B - x, \qquad V_1 = V, \qquad T_1 = T.$$

A burn of y base tokens followed by a top-up $N \ge 0$ yields

$$A_2 = A_1 + N = A + \frac{(A+V)x}{B-x} + N, \qquad B_2 = B - x - y, \qquad T_2 = T - y.$$

To optimize the virtual reserve to fully utilize the top-up amount as described in Section 2.3, the virtual reserve is adjusted to

$$V_2 = \frac{A_2 B_2}{T_2 - B_2} = \frac{A_2(B-x-y)}{T-B+x}. \tag{35}$$

When the attacker sells the x base tokens back, the pre-fee quote token outflow S_{pool} is determined by invariance:

$$(A_2 + V_2)B_2 = (A_2 - S_{\text{pool}} + V_2)(B_2 + x).$$

Solving and using Equation (35) and $B_2 + x = B - y$ gives

$$S_{\text{pool}} = \frac{A_2 \cdot x \cdot (T-y)}{(B-y)(T-B+x)}.$$

After applying the symmetric fee, the attacker receives

$$S_{\text{sell}} = \frac{1}{q} S_{\text{pool}} = \frac{x(T-y)}{q(B-y)(T-B+x)} \left(A + N + \frac{(A+V)x}{B-x} \right).$$

The net round-trip profit with top-up N is therefore

$$\Pi_{\text{topup}}(x, y, N) = S_{\text{sell}} - S_{\text{buy}} = \frac{x(T-y)}{q(B-y)(T-B+x)} \left(A + N + \frac{(A+V)x}{B-x} \right) - q\,\frac{(A+V)x}{B-x}.$$

Since Π_{topup} is increasing in N, the worst case for the protocol is the maximal top-up $N = M$ induced in Section 2.3. Substituting $N = M$ from Equation (12) gives

$$\begin{aligned}\Pi_{\text{topup}}(x, y, M) &= \frac{x(T-y)}{q(B-y)(T-B+x)} \cdot \frac{T-y}{T} V \cdot \frac{T-B+x}{B-x-y} - q\,\frac{(A+V)x}{B-x} \\ &= \frac{x(T-y)^2 V}{qT(B-y)(B-x-y)} - q\,\frac{(A+V)x}{B-x}.\end{aligned} \tag{36}$$

□

Theorem 2.1 (Safety under capped burns and top-ups)**.** Let there be a CBMM pool in state (A, B, V) with invariant $k = (A+V)B$, total base token supply $T > B$ and symmetric quote-token fee $N\%$ and fee multiplier $q = 100/(100-N) > 1$. Fix a burn cap parameter $\eta \in (0,1)$ and require that each burn event satisfies

$$0 < y \le \eta\,(B-x). \tag{37}$$

If

$$q \;\ge\; \frac{1}{1-\eta}, \tag{38}$$

then for all $A > 0$, $V > 0$, $B > 0$, $0 < x < B$, and all admissible burns y obeying Equation (37), the worst-case profit $\Pi_{\text{topup}}(x, y, M)$ from Lemma 2.1 is non-positive.

Proof. Starting from Equation (34), the burn cap Equation (37) implies

$$B - x - y \;\ge\; B - x - \eta(B-x) = (1-\eta)(B-x),$$

so

$$\frac{1}{B-x-y} \;\le\; \frac{1}{(1-\eta)(B-x)}.$$

Thus the first term in Equation (34) is bounded above by

$$\frac{x(T-y)^2 V}{qT(B-y)(1-\eta)(B-x)}.$$

Using $(T-y)^2 \le T^2$ (since $0 \le y < T$), we obtain the upper bound

$$\begin{aligned}\Pi_{\text{topup}}(x, y, M) &\le \frac{xT^2 V}{qT(B-y)(1-\eta)(B-x)} - q\,\frac{(A+V)x}{B-x} \\ &= \frac{xTV}{q(B-y)(1-\eta)(B-x)} - q\,\frac{(A+V)x}{B-x}.\end{aligned} \tag{39}$$

Factoring out the positive quantity $x/(B-x)$, we see that $\Pi_{\text{topup}}(x, y, M) \le 0$ is guaranteed whenever

$$\frac{TV}{q(B-y)(1-\eta)} - q(A+V) \;\le\; 0, \tag{40}$$

i.e.

$$TV \;\le\; q^2(A+V)(B-y)(1-\eta). \tag{41}$$

Substituting $TV = (A+V)B$ from $k = (A+V)B = VT$ and cancelling $A + V > 0$ yields

$$B \;\le\; q^2(B-y)(1-\eta). \tag{42}$$

For any admissible burn we have $y \le \eta(B-x) \le \eta B$, hence $B - y \ge (1-\eta)B$. Since the right-hand side of Equation (42) is increasing in $(B-y)$, it is enough to check the worst case $B - y = (1-\eta)B$, which gives

$$B \;\le\; q^2(1-\eta)^2 B \quad\iff\quad q^2(1-\eta)^2 \;\ge\; 1.$$

This is exactly Equation (38), i.e. $q \ge 1/(1-\eta)$. Under this condition we have $\Pi_{\text{topup}}(x, y, M) \le 0$ for all admissible (A, B, V, x, y), completing the proof. □

Parameter choice. Theorem 2.1 is stated for a generic per-burn cap η and fee multiplier q. In terms of the fee rate $N\%$, the condition Equation (38) becomes

$$\frac{100}{100 - N} \;\geq\; \frac{1}{1-\eta} \qquad \Longleftrightarrow \qquad N \;\geq\; 100\,\eta.$$

Thus any symmetric quote-token fee of at least $100\eta\%$ suffices to make the buy–burn–topup–sell loop unprofitable, regardless of the pool state and attack size, as long as each burn obeys the cap from Equation (37). This yields that if the burn percentage is less than the trading fee percentage, the loop is unprofitable.

3 Implementation

This section describes the CBMM implementation as an on-chain program. CBMM trading follows the same logic as a standard CPMM, so we focus on Continuous Conditional Buybacks (CCB), which offset the required reduction in the virtual reserve V after burns, followed by the burn safety controls and key configuration considerations. The design objective is to minimize user friction while fully preserving the guarantees established in Section 2.

We are describing Solana implementation, but most of the concepts are general and can be applied to other chains. For the initial version of the CBMM pool we have decided not to use a simple on-chain counter instead of an existing standard for the base token (like SPL or Token Extensions). This keeps the system closed and avoids external side effects. The quote token is a standard SPL token, and we keep using the “base token” and “quote token” terminology for consistency with the Section 2.

The implementation concepts described in this section are theoretical and might not directly reflective of the actual implementation details.

3.1 Continuous Conditional Buybacks

As discussed in Section 2.2, burns require a reduction of the virtual reserve from V_1 to V_2 (with $V_2 < V_1$). This adjustment reduces the positive impact of the burn. We implement the quote token reserve top-up mechanism described in Section 2.3 as **Continuous Conditional Buybacks** (CCB) redirecting a portion of trading fees into the Real Token Reserve A in real time.

Let $\Delta V = V_1 - V_2 \geq 0$ denote the required reduction in virtual reserve implied by Section 2. The implemented CCB mechanism then works as follows:

- **Fee accumulation**: On each trade, a fixed fraction of token fees is routed to a dedicated on-chain fee vault (an associated token account controlled by the program).
- **Burn-time top-up**: Upon a burn event, the program strives to top-up the pool real and virtual quote token reserves to the target values A_{opt} and V_{opt} as described in Section 2.3.
- **Continuous repayment**: While $A < A_{\text{opt}}$ and $V < V_{\text{opt}}$, fees contributed by the subsequent trades are immediately applied to increase A and V until they reach the target values. Any overage remains in the fee vault and is handled by the Fee accumulation rule.

The Continuous repayment step differs for base token buys and sells. For buys the fees are applied before the operation itself, for sells the fees are applied after the operation. This ensures that the potential partial top-up impact is maximized.

Technically, this is not a buyback, as no base tokens leave the pool. However, adding tokens to A increases the base token's price and effectively buys back part of the burn's price impact.

If not enough off-chain activity is observed, the trading fees are still accumulated and can be used to top-up the pool in the future. This motivates the off-chain activity to be persistent and ongoing.

3.2 Burn Safety Mechanisms

As discussed in the Section 2.4.1, the only way to keep the CBMM pool safe from exploitation is to make sure burns and fees are coordinated so nobody can extract collateral through a buy–burn–topup–sell loop. The risk doesn't only apply to a single burn, but we need to consider multiple chained burns as well if there is no market reaction to the individual burns. The mechanisms described in this section retain the original intent of the design – burns exist, but they are paced, sized, and authorized to help the CBMM pool state remain inside the provably safe region.

As previously stated in the Section 2.4.3, the worst-case profit from the buy–burn–topup–sell happens when $N = M$. For this to happen the trading preceding the burn must be high enough to fill the dedicated fee vault to the amount larger or equal to M. All the mechanism described in this section are either designed to lower the probability of the full top-up happening or directly protecting the system against the buy–burn–topup–sell loop.

3.2.1 Predictable burn size and burn cap

To make the impact of each burn simple to reason about, we restrict burns to a single predefined percentage of the pool. The program multiplies this percentage by the current base token reserve to obtain the burn amount y. This keeps every burn reasonably small, predictable, and directly comparable across different pool states.

As established in the mathematical model (see Theorem 2.1 and the subsequent parameter discussion), safety requires that the per-burn percentage does not exceed the symmetric trading fee percentage. In practice, we therefore choose a fixed burn fraction that is at most equal to the fee rate, typically smaller to retain additional safety margin.

3.2.2 CCB-first top-ups

The Continuous Conditional Buybacks logic described in Section 3.1 has a positive impact on the top-up size after burn. The fees gathered on every trade are immediately applied to increase the real quote token reserve to get to the optimal value (directly impacting the quote token virtual reserve as well), so the pool continuously drifts back toward the optimal state before the next burn.

The immediate“full” top-up after a burn (from Section 2.3) only triggers when $A = A_{\text{opt}}$ and $V = V_{\text{opt}}$ and the fee vault already holds M, preventing a burn from executing ahead of the capital needed to reduce V. If trading volume never fills the vault, the loop defaults to the simpler buy–burn–sell case analyzed earlier, which is inherently easier to defend.

3.2.3 Burn authority

If off-chain control of the user activity is needed we can implement a centralized Burn Authority that is responsible for approving burns. Every burn instruction must include a signature from this authority, which certifies that the burn is tied to a concrete off-chain action and that the caller is not only calling the on-chain program directly. This is not mandatory and can be turned off if abuse is not of a concern and there is more focus on the full decentralization.

3.2.4 Throttling mechanism

Because attacks can consist of several burns chained together, we rate-limit how many burns may be executed for a pool within a given time window. This gives honest traders time to react, forces bots to spread their attempts out in time, and makes it much harder to pre-script a profitable sequence. There are two possible approaches to handling the throttle sarutartion:

- **Reject excess burns:** Subsequent burn instructions are rejected until the window resets. This is the most straightforward approach, but can interfere with the user experience, as burns following an off-chain action might be rejected.
- **Queue and defer burns:** Burn amounts are queued and executed later when the throttle window is reset. This approach is more flexible and improves user experience, but it requires a more complex implementation.

If the Burn Authority from Section 3.2.3 is employed, the throttling logic is fully customizable off-chain and the on-chain logic can fe fully skipped.

3.2.5 Daily burn limits

To be able to restrict the number of impactful off-chain events (and accompanying burns) per user we keep per-wallet daily burn limits in a lightweight on-chain account, `UserBurnAllowance`, keyed by the wallet address. It records the burn count over the past 24 hours plus the timestamp of the most recent burn. Creation is permissionless (same pattern as e.g. SPL Associated Token Accounts) so anyone can fund [6] the account for a given user. If an account sits idle for ≥ 24 hours it can be closed and the rent returned to the original funding wallet. Once a user hits the configured ceiling, more burns are rejected until the window resets.

However, this doesn't protect against Sibyl attacks and is mainly implemented as a convenience mechanism to be able to track the faithful users and restrict their amount of impactful off-chain activity.

The proposed mechanism can be used to restrict activity across all pools on the platform as it is not specific to a single pool. If specificity is required, a separate account can be used for each pool, which would incur additional funding costs. To minimize these costs and better scalability, equivalent functionality could be realized via State Compression using Concurrent Merkle Trees [7], at the cost of off-chain infrastructure dependence.

3.3 Pool Configuration Considerations

The main decision that influences the pool behavior is the initial virtual reserve V. This reserve directly sets the token initial price and is proportional to the trading volume needed to be able to top-up the pool fully. Moreover, the higher initial reserve the more expensive it is to snipe a large amount of base tokens at the starting price.

4 Conclusion

This work set out to design a market-making mechanism that can translate verifiable off-chain activity into on-chain price impact through controlled burns. We constructed a mathematical model for CBMM with a virtual reserve V, derived closed-form expressions for trading and burning, and used them to characterize the induced price impact and worst-case exit price. On the implementation side, we outlined a Solana-oriented program architecture that realizes these mechanics with a closed base-token design (no external SPL mint), Continuous Conditional Buybacks (CCB) that route fees into the real quote reserve A, and a set of safety controls that coordinate burns, fees, and top-ups.

The safety analysis reveals a sharp separation between price mechanics and collateral extraction. Without fees, any real reserve $A > 0$ makes the buy–burn–sell loop always at least break-even and allows an attacker to extract almost all of A as $x, y \to B$. Adding symmetric quote-token fees but no top-ups yields a closed-form burn threshold $y_{\max}$ under which the loop is unprofitable, independently of the attacker's buy size, and we give a concrete numerical example showing that once maximal top-ups are allowed, fees alone no longer suffice: profitable buy–burn–topup–sell loops reappear even at nontrivial fee rates. We therefore derive an explicit worst-case profit formula with top-ups and prove a safety theorem: if each burn is capped to a fraction η of the pool and the symmetric fee multiplier $q = 100/(100 - N)$ satisfies $q \geq 1/(1 - \eta)$ (equivalently, the fee rate obeys $N \geq 100\eta\%$), then the loop is non-profitable for all pool states and attack sizes.

Operationally, this condenses into a simple rule: the per-burn fraction must not exceed the symmetric trading fee percentage, and is preferably smaller for additional margin. Mathematically, burns both raise price and force a downward adjustment of V; we show that after a burn there is a unique "optimal" target state $(A_{\text{opt}}, V_{\text{opt}})$ and top-up amount M that restore the original worst-case exit price, and we analyze how delayed and partial top-ups, combined with intervening trades, affect the required collateral and convergence toward this state. The CCB mechanism implements this path-dependently: fees continuously drift the pool toward $(A_{\text{opt}}, V_{\text{opt}})$, and a full top-up to M is only applied when the fee vault is sufficiently funded, which limits attacker access to fresh collateral.

To make the system robust in practice, we propose predictable fixed-percentage burns (a single configured fraction of the current base reserve, bounded by the fee rate), an optional Burn Authority tying burns to off-chain actions, time-based throttling of burn frequency (reject-or-queue semantics), and per-wallet daily burn limits via a lightweight on-chain `UserBurnAllowance` account (optionally generalized via compressed state). These mechanisms reduce the probability that the worst-case top-up scenario can be orchestrated repeatedly, and they remain compatible with fully decentralized configurations when centralized authority is disabled.

This paper deliberately focuses on CBMM as a base token rollout mechanism with $A_0 = 0$ and a constant-product invariant, but several extensions are natural directions for future work: generalizing CBMM to other asset pairs and nonzero initial quote token reserves, introducing third-party liquidity provisioning, and investigating CBMM pools as a part of a broader financial ecosystem where cross-pool arbitrage, routing, and price discovery interact with the burn and top-up logic. Looking into alternative invariant families such as Power CBMM with $k = (A + V)B^p$ for $p > 0$ is another natural direction. These invariants allow shaping the bonding curve (e.g., $p > 1$ to front-load price impact and mitigate sniping, or $0 < p < 1$ for flatter curves) but make virtual-reserve adjustment and safety proofs substantially more complex, and they significantly complicate the on-chain integer-arithmetic implementation. A more systematic treatment of such generalized invariants is also left for future study.

References

[1] Guillermo Angeris and Tarun Chitra. “Improved Price Oracles: Constant Function Market Makers”. In: *arXiv preprint arXiv:2003.10001* (2020). arXiv: `2003.10001 [q-fin.TR]`. URL: `https://arxiv.org/abs/2003.10001`.

[2] Hayden Adams. *Uniswap Whitepaper*. 2020. URL: `https://hackmd.io/@HaydenAdams/HJ9jLsfTz`.

[3] Hayden Adams et al. *Uniswap v3 Core*. `https://app.uniswap.org/whitepaper-v3.pdf`. Mar. 2021.

[4] Meteora Team. *What’s DAMM v2? (Product Overview)*. Accessed Nov 5, 2025. 2025.

[5] Eyal Hertzog, Guy Benartzi, and Galia Benartzi. *Bancor Protocol: Continuous Liquidity and Asynchronous Price Discovery for Tokens through their Smart Contracts (aka “Smart Tokens”)*. `https://resources.bancor.network/pages/BancorProtocolWhitepaper.pdf`. Describes the constant-reserve-ratio bonding curve mechanism; archived translations available on GitHub. 2018.

[6] *Rent on Solana*. `https://docs.solana.com/developing/programming-model/accounts#rent`. 2022.

[7] Jarry Xiao et al. *Compressing Digital Assets with Concurrent Merkle Trees*. Whitepaper. Concurrent Merkle Tree whitepaper; accessed 2025-11-06. Solana Labs, 2023. URL: `https://drive.google.com/file/d/1BOpa5OFmara5OfTvL0VIVYjtg-qzHCVc/view`.