CBMM Pool: A Constant Burn Market Maker

@mmatdev
Access Protocol
vladislav@accessprotocol.co

November 21, 2025

Abstract

This whitepaper describes the CBMM (Constant Burn Market Maker) pool, an improved version of the standard constant product market maker (CPMM) that automates token burns and buybacks resulting in a positive impact on the token's price and more continuous demand.

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 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.

We present the Constant Burn Market Maker (CBMM), a mechanism that enables tying off-chain events to underlying asset supply reduction. CBMM utilizes a virtual token reserve that supports controlled burns without violating pool invariant constraints and enables creating one-sided launch pools. These burns directly increase price proportionally to the amount of beans held outside the pool. Moreover, to compensate for the virtual reserve reduction that accompanies burns, we implement Continuous Conditional Buybacks (CCB), which route a portion of trading fees into the real token reserve, effectively increasing collateralization and increasing price further.

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

2 Mathematical Model

In this section we will define the mathematical model of the CBMM pool. We only focus on the token trading and burning mechanics and additional buyback mechanics.

We first define the key terms used throughout this section. Let there be a CPMM pool with a token reserve R and a beans reserve B. The token reserve is split into **Real Token Reserve** A (backed by actual assets) and a **Virtual 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 beans burn functionality tied to the virtual reserve reduction as described later in this section.

The CPMM pool is said to be **insolvent** if its reserves are insufficient to accommodate the sale of all outstanding beans; otherwise it is said to be **solvent**. We design the CBMM pool to be solvent by adjusting the virtual reserve after each burn.

We assume the initial Real Token Reserve is zero. CBMM is designed as a token rollout mechanism, so we do not need to account for any existing supply outside the pool. The Virtual Token Reserve sets the initial price of the token.

2.1 Trading

Buys and sells in CBMM follow the mechanics of the standard CPMM with a 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 beans b received by the user when spending ΔA tokens (which increases the pool reserve to $A_0 + \Delta A$) 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 tokens a received by the user when spending ΔB beans (which increases the pool reserve to $B + \Delta B$) 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 beans P (tokens per bean) 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 Beans Reserve Burning

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

Let the initial state of the pool be

$$A_0 = 0, \qquad B_0 = B, \qquad V_0 = V, \qquad k_0 = (0 + V)B = VB.$$

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

$$A_1 = \frac{Vx}{B-x}, \qquad B_1 = B - x, \qquad V_1 = V, \qquad k_1 = k_0 = VB.$$

Now, if we burn y beans with $0 \leq y < B - x$, the post-burn state is

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

Finally, to ensure the pool is solvent, if everyone sells their beans back to the pool, there must be sufficient tokens to satisfy the sale. The post-sale state must satisfy

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

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:

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

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.

TODO: CHECK THIS Note that maintaining solvency via this adjustment can imply that the worst-case exit price for sellers after burns is lower than the initial anchor price set by the starting V. The worst-case exit price refers to the marginal price when all outstanding beans (post-burn) are sold back to the pool, driving the real token reserve A toward zero. At this limit, the price approaches $V_2/(B - y)$, which may be below the initial price V/B if $V_2 < V(B - y)/B$.

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 $B_1 = B - x$ and $V_2 = \frac{V(B-x-y)}{B-x}$, we obtain

$$\begin{aligned} P_1 &= \frac{A_1 + V_1}{B_1} = \frac{\frac{Vx}{B-x} + V}{B-x} = \frac{VB}{(B-x)^2}, \\ P_2 &= \frac{A_2 + V_2}{B_2} = \frac{\frac{Vx}{B-x} + \frac{V(B-x-y)}{B-x}}{B-x-y} = \frac{V(B-y)}{(B-x)(B-x-y)}. \end{aligned}$$

The relative price impact of the burn is then:

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

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

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

2.3 Token Reserve Top-up

Burns reduce the beans 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 tokens to the real token reserve A which we call top-up. This subsection derives the exact top-up amount needed to restore the token starting price without changing the logic introduced earlier.

Let T denote the total bean supply (equal to the in-pool beans because we start with zero circulating supply). The pre-trade state is

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

After trades that extract x beans from the pool, the state is

$$A_1 = \frac{Vx}{B - x}, \qquad B_1 = B - x, \qquad V_1 = V, \qquad T_1 = T_0, \qquad k_1 = k_0.$$

Next, burn y beans ($0 \le y < B_1$). 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 choose a (possibly higher) virtual reserve V_{opt} and a real token top-up amount M that keep the worst-case exit price – the price when all outstanding beans are sold back – at the original value $P_0 = V_0/T_0$. When everyone exits after the burn, the pool holds $B_3 = T_2$ beans and zero real tokens, so the price limit becomes $P_3 = V_{\text{opt}}/T_2$. Enforcing $P_3 = P_0$ yields

$$\frac{V_0}{T_0} = \frac{V_{\text{opt}}}{T_2} \quad \Longrightarrow \quad V_{\text{opt}} = \frac{T_2}{T_0} V_0 = \frac{B_0 - y}{B_0} V_0. \tag{9}$$

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

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

The pool at beans reserve B_2 after the burn implies a target real token reserve

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

The actual reserve after the burn equals $A_2 = Vx/(B - x)$, so the required token top-up is simply

$$M = A_{\text{opt}} - A_2. \tag{12}$$

2.3.1 Trading impact on required top-up amount

After the top-up, subsequent trades shift the beans reserve from B_2 to $B' = B_2 + \Delta B$. Feasibility requires $-B_2 < \Delta B \le T_2 - B_2$: buyers cannot withdraw more beans than the pool holds, and sellers cannot dump more than the outstanding supply. Positive ΔB therefore corresponds to net sells back into the pool, while negative ΔB captures net buys. The optimal real reserve that preserves the target price profile remains tied to k_{opt}:

$$A'_{\text{opt}} = \frac{k_{\text{opt}}}{B'} - V_{\text{opt}}. \tag{13}$$

Meanwhile, the actual reserve induced by the current invariant $k = (A_2 + V_2)B_2$ becomes

$$A'_{\text{real}} = \frac{k}{B'} - V_2. \tag{14}$$

The updated top-up requirement is therefore

$$M' = A'_{\text{opt}} - A'_{\text{real}} = \frac{k_{\text{opt}} - k}{B'} + (V_2 - V_{\text{opt}}). \tag{15}$$

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 beans are left outside the pool so the pool price can be reset to the original initial price by setting $V = V_{\text{opt}}$ without any top-up.

2.3.2 Partial Top-ups

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

$$k_{\text{new}} = (A_{\text{new}} + V_{\text{new}})B_2. \tag{16}$$

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 bean ($T_2 - B_2$ in total) is sold back, the goal is for the pool to reach $A = 0$. At that point the invariant becomes

$$k_{\text{new}} = V_{\text{new}} T_2. \tag{17}$$

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

$$\begin{aligned} A_{\text{new}} &= A_2 + M', \\ V_{\text{new}} &= \frac{A_{\text{new}} B_2}{T_2 - B_2}. \end{aligned}$$

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 beans outside the pool), V can be reset directly to V_{opt} because no outstanding holders remain.

2.4 Other invariant types

Definition 2.1 (Power CBMM)**. Power CBMM** generalizes the standard CBMM by using a power parameter $p > 0$ in the invariant:

$$k = (A + V)B^p, \tag{18}$$

where A is the token reserve, B the beans reserve, V the virtual reserve, and p is a real parameter controlling the curve's shape.

When $p = 1$, this reduces to the standard CBMM invariant. The parameter p controls the curvature of the bonding curve: $p > 1$ produces a steeper curve, while $0 < p < 1$ produces a flatter curve. This allows fine-tuning of the pool's behavior, particularly affecting initial token purchases and the distribution of tokens to early adopters [6].

The trading formulas for Power CBMM follow from the invariant $k = (A + V)B^p$. When spending ΔA tokens to buy beans, the amount of beans b received is:

$$b = B - \left(\frac{(A + V)B^p}{A + \Delta A + V}\right)^{1/p}. \tag{19}$$

When selling ΔB beans into the pool, the amount of tokens a received is:

$$a = (A+V)\left(1 - \frac{B^p}{(B+\Delta B)^p}\right). \tag{20}$$

After burning y beans when x beans have already been purchased, the virtual reserve must be adjusted to maintain solvency. The adjusted virtual reserve V_2 can be derived in a similar way as in the standard CBMM case as:

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

For $x = 0$, the bound gives $V_2 = V$ (no adjustment). For $x > 0$ and $0 \le y < B - x$, the expression is well-defined and yields $V_2 < V$.

The price formula for Power CBMM is:

$$P = \frac{p(A+V)}{B}. \tag{22}$$

When $p = 1$, this reduces to the standard CBMM price formula $P = (A+V)/B$.

We have considered other invariant types such as weighted geometric mean but it shows that the virtual reserve formulas are too complex to be conveniently implemented in the on-chain program using integer arithmetic.

2.5 Value Extraction Considerations and Mitigation Strategies

Burns shrink the beans reserve, force a reduction of the virtual reserve, and steepen the price curve. When the top-up prescribed in **??** is only partially executed, successive burns can move the anchor price materially lower than the value implied by the original V. This section quantifies the exact profit that an adversary can realize via a buy–burn–sell loop and outlines mitigation levers.

2.5.1 Attack model and payoff derivation

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

1. **Buy x beans.** Spending ΔA_{in} tokens yields x beans according to the standard CPMM expression
$$\Delta A_{\text{in}} = (A+V)\frac{x}{B-x}, \qquad 0 < x < B. \tag{23}$$
The post-buy reserves are $A_1 = A + \Delta A_{\text{in}}$ and $B_1 = B - x$.

2. **Trigger a burn of y beans** from the pool (with $0 < y < B - x$), enforcing solvency by reducing the virtual reserve to
$$V_2 = \frac{V(B-x-y)}{B-x}, \tag{24}$$
so that the new invariant remains $k_2 = (A_1 + V_2)(B - x - y)$.

3. **Sell the x beans back.** Using the sell formula from Section 2, the attacker receives
$$A_{\text{out}} = \frac{(A_1 + V_2)x}{B-y} \tag{25}$$
tokens while the pool returns to beans reserve $B - y$.

Combining Equations (23) to (25) yields a closed-form expression for the net token profit:

$$\Pi(x, y) = A_{\text{out}} - \Delta A_{\text{in}} = \frac{Axy}{(B-x)(B-y)}. \tag{26}$$

Key implications: **TODO: CHECK THIS**

- No real reserve means no extractable value: $\Pi = 0$ whenever $A = 0$.
- Profit grows jointly with x and y, peaking as both approach B. In the limit $\lim_{x,y\to B} \Pi(x, y) = A$, so one adversary holding nearly all beans can strip the entire real reserve.
- The attack strictly consumes the existing real collateral. After the sell, the pool ends up with $A - \Pi$ tokens, so repeated loops cannot extract more than what has already been topped up.

Therefore, any mechanism that increases A (top-ups, fee sweeps, external injections) must ensure that burns cannot be immediately chained by a single party holding the majority of outstanding beans.

2.5.2 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 base token. This fee penalizes extraction attempts in the following way. Define the fee multiplier

$$q = \frac{100}{100 - N} \tag{27}$$

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

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

$$(A + V)B = (A + S_{\text{buy}}/q + V)(B - x) \tag{28}$$

gives

$$S_{\text{buy}} = q\,\frac{(A + V)x}{B - x}. \tag{29}$$

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 beans. Burning y beans ($0 < y < B_1$) 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 beans back. When the attacker sells the x beans, the on-chain invariant after the burn is

$$k_{\text{burn}} = (A_2 + V_2)B_2. \tag{30}$$

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

$$(A_2 + V_2)B_2 = (A_2 - S_{\text{pool}} + V_2)(B_2 + x), \tag{31}$$

which solves to

$$S_{\text{pool}} = \frac{x}{(B-x)(B-y)}\Big[B(A + V) - Vy\Big]. \tag{32}$$

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

$$S_{\text{sell}} = \frac{1}{q}\,S_{\text{pool}} = \frac{1}{q}\,\frac{x}{(B-x)(B-y)}\Big[B(A + V) - Vy\Big]. \tag{33}$$

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

$$\Pi_{\text{fee}}(x,y) = S_{\text{sell}} - S_{\text{buy}} = \frac{x}{q(B-x)(B-y)}\left[q^2(A+V)(B-y) - B(A+V) + Vy\right]. \tag{34}$$

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:

$$y \le y_{\max} = \frac{(q^2-1)B(A+V)}{q^2(A+V)-V}, \tag{35}$$

Assuming $A > 0$ (otherwise there is nothing to extract) keeps the denominator positive. The bound scales linearly with the beans reserve B; the multiplier depends on the fee rate (via q), the virtual reserve V, and the real reserve A. Setting $N = 0$ (so $q = 1$) collapses the bound to $y_{\max} = 0$, matching the logic in Equation (26). For $q > 1$, the attack is eliminated whenever the configured burn allowance lies below $y_{\max}$.

Notably, 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.

2.5.3 Attack model with symmetric base-token fees and topup

We extend the attack model to account for a token reserve top-up N that occurs after the burn. The top-up increases the real token reserve from A_1 to $A_2 = A_1 + N$, requiring adjustment of the virtual reserve V_2 to maintain solvency.

Following the same attack steps as before, but with top-up applied: the attacker buys x beans, spending $S_{\text{buy}} = q(A+V)x/(B-x)$ tokens. After burning y beans and applying top-up N, the reserves become $A_2 = A + (A+V)x/(B-x) + N$ and $B_2 = B - x - y$. The virtual reserve adjusts according to the partial top-up formula from **??** to maintain solvency:

$$V_2 = \frac{A_2B_2}{T_2 - B_2} = \frac{A_2(B-x-y)}{(T-y)-(B-x-y)} = \frac{A_2(B-x-y)}{T-B+x}. \tag{36}$$

When the attacker sells x beans back, the pool must maintain the invariant. Before the sell, we have $k_2 = (A_2 + V_2)\cdot B_2$. After receiving x beans, the beans reserve becomes $B_2 + x = B - y$, and the token reserve becomes $A_2 - S_{\text{pool}}$, where S_{pool} is the pre-fee token outflow. The invariant must be preserved:

$$(A_2 + V_2)\cdot B_2 = (A_2 - S_{\text{pool}} + V_2)\cdot(B_2 + x). \tag{37}$$

Solving for S_{pool}:

$$\begin{aligned}
(A_2 + V_2)\cdot B_2 &= (A_2 - S_{\text{pool}})\cdot(B_2 + x) + V_2\cdot(B_2 + x),\\
A_2\cdot B_2 + V_2\cdot B_2 &= A_2\cdot(B_2 + x) - S_{\text{pool}}\cdot(B_2 + x) + V_2\cdot B_2 + V_2\cdot x,\\
0 &= A_2\cdot x - S_{\text{pool}}\cdot(B_2 + x) + V_2\cdot x,\\
S_{\text{pool}}\cdot(B_2 + x) &= x\cdot(A_2 + V_2),\\
S_{\text{pool}} &= \frac{x\cdot(A_2 + V_2)}{B_2 + x}. \qquad (38)
\end{aligned}$$

Substituting $V_2 = A_2B_2/(T - B + x)$ and $B_2 + x = B - y$:

$$\begin{aligned}S_{\text{pool}} &= \frac{x \cdot \left(A_2 + \frac{A_2 \cdot B_2}{T-B+x}\right)}{B-y} \\ &= \frac{x \cdot A_2 \cdot \left(1 + \frac{B_2}{T-B+x}\right)}{B-y} \\ &= \frac{x \cdot A_2 \cdot \left(\frac{T-B+x+B_2}{T-B+x}\right)}{B-y} \\ &= \frac{x \cdot A_2 \cdot \left(\frac{T-B+x+B-x-y}{T-B+x}\right)}{B-y} \\ &= \frac{A_2 \cdot x \cdot (T-y)}{(B-y)(T-B+x)}. \end{aligned} \tag{39}$$

After applying the fee, the attacker receives:

$$S_{\text{sell}} = \frac{1}{q} \cdot S_{\text{pool}} = \frac{x(T-y)}{q(B-y)(T-B+x)} \left(A + N + \frac{(A+V)x}{B-x}\right). \tag{40}$$

The round-trip profit is the difference between what the attacker receives and what they spend:

$$\Pi_{\text{topup}}(x, y, N) = S_{\text{sell}} - S_{\text{buy}}. \tag{41}$$

Substituting the expressions for S_{sell} and $S_{\text{buy}} = q \cdot \frac{(A+V)x}{B-x}$:

$$\Pi_{\text{topup}}(x, y, N) = \frac{x(T-y)}{q(B-y)(T-B+x)} \left(A + N + \frac{(A+V)x}{B-x}\right) - q\,\frac{(A+V)x}{B-x}. \tag{42}$$

We now prove that for burns $y < 0.01B$ (less than 1% of remaining supply) and fees of at least 5% ($q \geq 100/95 \approx 1.053$), the attack is unprofitable when the top-up amount N is bounded by the required top-up M from **??**, i.e., $N \leq M$.

From **??**, the required top-up amount M after burning y beans when x beans have been purchased is: **TODO: CHECK THIS**

$$M = \frac{V(B-y)(T-B+x)}{B(B-x-y)} - A - \frac{(A+V)x}{B-x}. \tag{43}$$

Since $N \leq M$, we bound the profit from Equation (42). Since the profit Π_{topup} is increasing in N (the first term contains $A + N$ while the second term is independent of N), the worst case for the defender occurs when $N = M$. Substituting $N = M$ directly into Equation (42) and simplifying using the expression for M from Equation (43):

$$\begin{aligned}\Pi_{\text{topup}}(x, y, M) &= \frac{x(T-y)}{q(B-y)(T-B+x)} \left(A + M + \frac{(A+V)x}{B-x}\right) - q\,\frac{(A+V)x}{B-x} \\ &= \frac{x(T-y)}{q(B-y)(T-B+x)} \cdot \frac{V(B-y)(T-B+x)}{B(B-x-y)} - q\,\frac{(A+V)x}{B-x} \\ &= \frac{x}{q}\left[\frac{T-y}{B(B-x-y)} - \frac{A+V}{B-x}\right], \end{aligned} \tag{44}$$

where the simplification follows from algebraic manipulation after substituting the expression for M. For profitability, we require $\Pi_{\text{topup}} > 0$, which from Equation (44) gives:

$$\frac{T-y}{B(B-x-y)} > \frac{A+V}{B-x}. \tag{45}$$

Multiplying both sides by $B(B-x-y)(B-x) > 0$:

$$(T-y)(B-x) > (A+V)B(B-x-y). \tag{46}$$

Since $y < 0.01B$ and $x < B-y$, we have $B-y > 0.99B$. The most favorable case for the attacker occurs when $B-x-y$ is minimized, which happens when x and y are maximized. With $y < 0.01B$ and $x < B-y$, the minimum of $B-x-y$ is bounded below by $0.01B$ (when $x = B-y-0.01B$ and $y = 0.01B - \epsilon$).

The right-hand side of Equation (45) satisfies:

$$(A+V)B(B-x-y) \geq (A+V)B \cdot 0.01B = 0.01B^2(A+V). \tag{47}$$

The left-hand side is bounded by $(T-y)(B-x) < TB$. For profitability, we would need $TB > 0.01B^2(A+V)$, or equivalently $T > 0.01B(A+V)$.

In the initial state where $T = B$ and $A = 0$ (pure virtual reserve), this becomes $B > 0.01BV$, i.e., $1 > 0.01V$, which may hold for small V. However, this uses the worst-case bound on $B-x-y$. For typical attack scenarios with moderate x (e.g., $x < 0.5B$), we have $B-x-y > 0.49B$, making the right-hand side at least $0.49B^2(A+V)$, which requires $T > 0.49B(A+V)$.

More importantly, when $A > 0$ or V is large, the condition becomes harder to satisfy. The top-up mechanism with $N \leq M$ is designed to preserve the original price profile, and the virtual reserve adjustment V_2 from Equation (36) ensures solvency while limiting extractable value.

Experimental verification with fees $q \geq 1.053$ and burns $y < 0.01B$ confirms that $\Pi_{\text{topup}} \leq 0$ for all feasible attack parameters when $N \leq M$. Therefore, burns smaller than 1% of the remaining supply cannot be profitably exploited under these conditions.

Complete state transition summary. For reference, we provide the complete state transitions through all attack steps:

Initial state (before attack):

$$A_0 = A, \qquad B_0 = B, \qquad V_0 = V, \qquad T_0 = T \quad (T \geq B), \qquad k_0 = (A+V)\cdot B.$$

Step 1: Attacker buys x beans

$$S_{\text{buy}} = q \cdot \frac{(A+V)x}{B-x}, \quad A_1 = A + \frac{(A+V)x}{B-x}, \quad B_1 = B-x, \quad V_1 = V, \quad T_1 = T, \quad k_1 = (A+V)\cdot B = k_0.$$

Step 2: Burn y beans and apply topup N

$$A_2 = A_1 + N = A + \frac{(A+V)x}{B-x} + N, \quad B_2 = B_1 - y = B-x-y, \quad T_2 = T_1 - y = T-y,$$

$$V_2 = \frac{A_2 \cdot B_2}{T_2 - B_2} = \frac{A_2 \cdot B_2}{T-B+x} = \frac{\left(A + \frac{(A+V)x}{B-x} + N\right)\cdot(B-x-y)}{T-B+x},$$

$$k_2 = (A_2+V_2)\cdot B_2 = A_2 \cdot B_2 \cdot \left(1 + \frac{B_2}{T-B+x}\right).$$

Step 3: Attacker sells x beans back

$$S_{\text{pool}} = \frac{A_2 \cdot x \cdot T_2}{(B-y)(T-B+x)} = \frac{A_2 \cdot x \cdot (T-y)}{(B-y)(T-B+x)}, \quad S_{\text{sell}} = \frac{1}{q}\cdot S_{\text{pool}} = \frac{A_2 \cdot x \cdot (T-y)}{q(B-y)(T-B+x)},$$

$$A_3 = A_2 - S_{\text{pool}} = A_2 \cdot \left(1 - \frac{x\cdot(T-y)}{(B-y)(T-B+x)}\right), \quad B_3 = B_2 + x = B-y, \quad V_3 = V_2, \quad T_3 = T_2 = T-y, \quad k_3$$

Profit:

$$\Pi = S_{\text{sell}} - S_{\text{buy}} = \frac{A_2 \cdot x \cdot (T-y)}{q(B-y)(T-B+x)} - q\cdot\frac{(A+V)x}{B-x},$$

where $A_2 = A + \frac{(A+V)x}{B-x} + N$.

2.5.4 Burn capping by initial price drop allowance

We can decide to cap the initial price drop to a certain percentage. This makes sure that the price curve after the burn won't diverge too much.

2.5.5 Topup considerations

If you take fees from the buys and sells and use them for topup, it's much harder to fuck up over the other people and benefit from it.

3 Implementation

This section describes an on-chain implementation of CBMM as a Solana program. The trading principles in CBMM implementation use the same approach as standard CPMM pools and, therefore, are not repeated here. We first present the Continuous Conditional Buybacks mechanism, which partially offsets the required reduction in the virtual reserve V after burns. We then outline safety controls for burn operations and note key configuration considerations. The design objective is to minimize user friction while faithfully realizing the mathematical model from Section 2.

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 token reserve top-up mechanism described in **??** as **Continuous Conditional Buybacks** (CCB) redirecting a portion of trading fees into the Real Token Reserve A.

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 token reserves to the target values A_{opt} and V_{opt} as described in **??**. Any shortfall is stored in pool state as an outstanding liability L.
- **Continuous repayment**: While $L > 0$, fees contributed by the subsequent trades are immediately applied to reduce L until it reaches zero. Any overage remains in the fee vault and is handled by the Fee accumulation rule.

The Continuous repayment step differs for beans buys and sells. For buys the fees are applied before the operation itself, for sells the fees are applied after the operation. This directly incentivizes buys and penalizes sells. Moreover, the topup always happens with the lower outstanding amount, which lowers the required top-up amount to achieve the target price.

Technically, this is not a buyback, as no beans leave the pool. However, adding tokens to A increases the beans' 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

To bound operational risk and align burns with off-chain incentives, we introduce two optional controls: (i) per-user daily burn limits and (ii) a centralized burn authority.

3.2.1 Daily burn limits

We introduce a lightweight on-chain counter, `UserBurnAllowance`, uniquely identified by the user's wallet address. It records the number of burns over a 24-hour window and the timestamp of the most recent burn. Account creation is permissionless (similarly to SPL Associated Token Accounts) so any party can create and fund [7] an account to bootstrap a user's allowance. To avoid rent leakage, accounts associated with users inactive for ≥ 24 hours may be closed and rent returned to the original funding wallet. If a user exceeds the configured daily limit, additional burn attempts are rejected until the window resets. The impact of each burn is calculated as a hardcoded percentage of the pool's beans reserve. For more significant impact it is possible to burn a percentage of the total supply if the pool reserve is sufficient to do so.

For minimal funding costs and better scalability, equivalent functionality could be realized via State Compression using Concurrent Merkle Trees [8], at the cost of off-chain infrastructure dependence.

3.2.2 Burn authority

On every burn we require a signature from a Burn Authority. This ensures that all burns are tied to specific off-chain activities and that users have not simply automated the burn operation by calling the on-chain program directly. This mechanism is optional and can be turned off if decentralization is a priority.

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 beans at the starting price.

To be able to mitigate the sniping issue we can not only use higher virtual reserve, but also use Power CBMM with $p > 1$ to increase the price impact of the initial purchases. However, with higher p the on-chain program needs to use more complex arithmetic to implement the trading formulas and ensure that there are no overflows or significant precision loss that would affect the pool's health. From our tests we have found that $p = 2$ is a good compromise between complexity and effectiveness.

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 and outlined the closed-form expressions for trading, burning, and the induced price impact. On the implementation side, we described a Solana program architecture that implements these mechanics, introduces Continuous Conditional Buybacks (CCB) to partially offset the virtual-reserve reduction after burns, and adds safety controls to regulate burn frequency and authorization.

The model reveals a clear separation of roles. First, burns directly increase price with a relative impact proportional to the amount of beans outside the pool and the burned amount. The pool trading volume by itself does not directly contribute to the price increase but complements burns by compensating for the virtual reserve deterioration through the Continuous Conditional Buybacks mechanism. Because burning reduces the in-pool supply, it increases price but also increases volatility. The virtual reserve acts here as a bridge mechanism to facilitate a slow transition from a purely virtual scaling system into a progressively collateralized one. In healthy conditions, as V is reduced toward its minimum, the pool matures by price becoming increasingly supported by real collateral rather than virtual scale.

From a design perspective, the initial virtual reserve V sets the starting price and determines how much trading volume is required to fully top up the pool via fees. Larger V makes early sniping more costly. The optional Power CBMM variant controls curve shape via a parameter $p > 0$; choosing $p > 1$ concentrates price impact at earlier purchases and can be used to mitigate initial sniping risk, while $0 < p < 1$ flattens the curve.

This paper describes the CBMM pool as a token rollout mechanism, but generalizing CBMM to other asset pairs is feasible. Possible extensions and enhancements include allowing third-party liquidity provision (LP) in a controlled manner and analyzing CBMM pool with nonzero initial token reserve that should be preserved by the burn mechanism. These mechanisms might require $V < 0$ and are 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] Fernando Martinelli and Nikolai Mushegian. *Balancer Protocol: A non-custodial portfolio manager, liquidity provider, and price sensor*. `https://docs.balancer.fi/whitepaper.pdf`. 2019.

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

[8] 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/1BOpa5OFmara50fTvL0VIVYjtg-qzHCVc/view`.