CBMM Pool: A Constant Burn Market Maker

@mmatdev
Access Protocol
vladislav@accessprotocol.co

November 6, 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 pool assets 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, the buyback mechanics are an independent extension of this model and therefore will be discussed in Section 3.1.

We first define the key terms used throughout this section. Let there be a CPMM pool with a token reserve R and a bean supply 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. Denote the pre-trade (buy or sell) values as:

$$\begin{aligned} &A = A_0, \\ &B = B_0, \\ &V, \\ &k = (A_0 + V)B_0 \end{aligned}$$

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 can be computed at any time as:

$$P = \frac{A + V}{B}. \tag{5}$$

2.2 Beans Supply Burning

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

Let the initial state of the pool be:

$$
\begin{aligned}
A_0 &= 0, \\
B_0 &= B, \\
V_0 &= V, \\
k_0 &= (0 + V)N = VN
\end{aligned}
$$

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

$$
\begin{aligned}
A_1 &= \frac{Vx}{B - x}, \\
B_1 &= B - x, \\
V_1 &= V, \\
k_1 &= k_0 = VB
\end{aligned}
$$

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

$$
\begin{aligned}
A_2 &= \frac{Vx}{B - x}, \\
B_2 &= B - x - y, \\
V_2 &\text{ to be determined}, \\
k_2 &= (A_2 + V_2)B_2
\end{aligned}
$$

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:

$$
\begin{aligned}
A_3 &\geq 0, \\
B_3 &= B - y, \\
V_3 &= V_2, \\
k_3 &= (A_3 + V_3)B_3
\end{aligned}
$$

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:

$$
V_2 \leq \frac{V(B - x - y)}{B - x}. \tag{6}
$$

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.

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.

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:

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

The relative price impact of the burn is then:

$$\frac{P_2 - P_1}{P_1} = \frac{xy}{B(B-x-y)}. \tag{7}$$

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 Other invariant types

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

$$k = (A + V)B^p, \tag{8}$$

where A is the token reserve, B the bean supply, 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:

$$b = B - \left(\frac{(A+V)B^p}{A + \Delta A + V}\right)^{1/p}. \tag{9}$$

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

$$a = (A + V)\left(1 - \frac{B^p}{(B + \Delta B)^p}\right). \tag{10}$$

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:

$$V_2 = \frac{V\left((B-x)^p - B^p\right)(B-x-y)^p}{(B-x)^p\left((B-x-y)^p - (B-y)^p\right)}, \tag{11}$$

We have considered other invariant types 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.

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

Burns require a reduction of the virtual reserve from V_1 to V_2 (with $V_2 \leq V_1$). This adjustment reduces the positive impact of the burn. We implement **Continuous Conditional Buybacks** (CCB) to partially offset this by redirecting a portion of trading fees into the Real Token Reserve A, which bumps the price slightly back up.

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 deposits into A an amount $\Delta A = \min(\Delta V,\ F)$, where F is the available collected fee balance. Any shortfall $L = \Delta V - \Delta A_{\text{topup}} \geq 0$ is stored in pool state as an outstanding liability.
- **Continuous repayment**: While $L > 0$, subsequent trades continue to contribute fees that 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 choice of top-up target $V_2 - V_1$ is an arbitrary decision, other targets are possible. This decision has been made because it naturally "compensates" for the virtual reserve reduction, even though this compensation is not exact as the amount in the real token reserve can be later extracted while trading. 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 behaviour is the initial virtual reserve V. This reserve directly sets the token initial price and is linearly related 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 is 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 detoriation 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.