Zest CLMM

October 2026

Zest Protocol

ABSTRACT

Concentrated liquidity lets a provider commit capital to a bounded price range. Zest CLMM, an automated market maker for Stacks, partitions liquidity into a fixed geometric grid of bins, each with its own bounded constant-product curve and fungible shares. Liquidity and price are derived from a bin's reserves, retained fees compound into those reserves, and a deposit that changes a bin's composition is charged the swap fee on the exchange implied by adding and immediately redeeming liquidity. We present the model, prove its liquidity and valuation properties, and describe a Clarity implementation on 128-bit integers whose liquidity estimates err only downward, by at most 2^{-56} of the exact value plus one frame-liquidity unit.

1 INTRODUCTION

Concentrated liquidity allows a provider to allocate capital to a bounded price range. Within that range, virtual reserves give the position the trading depth of a larger constant-product pool. Uniswap v3 describes this construction for positions with individually chosen, tick-aligned bounds [1]. Because positions with different bounds are not interchangeable, v3 tracks each one separately and accumulates its fees outside the curve, where they do not compound. Liquidity Book instead partitions the price axis into bins with fungible shares, but each bin is a constant-sum curve with a fixed marginal price, so price moves only from one bin to the next [2]. Maverick v2 gives each tick a bounded constant-product curve with pro-rata claims on its reserves [10].

Zest CLMM applies bounded constant-product bins to a fixed geometric grid on Stacks. Providers in a bin share one inventory and hold fungible, proportional claims on it, so shares can be held by vaults and other contracts like any token. A position spanning several bins is a portfolio of these claims.

This paper makes the following contributions:

- *Reserve-derived bins* (Sections 3 and 4). For fixed bin bounds, a nonempty reserve pair determines a unique positive liquidity and a price in the interval, so neither is stored and retained fees compound into reserves. Section 2 compares this with Maverick v2.
- *Deposit fees* (Section 5). The fee on an imbalanced deposit is defined exactly, as the swap fee on the fee-free exchange implied by adding and immediately redeeming liquidity.
- *Conservative integer execution* (Section 6, Appendices B and C). On Clarity's 128-bit integers, liquidity estimates carry a proved one-sided error bound, and wide products use an exact multiword multiply–divide.

Section 6 also describes aggregate settlement of multi-bin routes, an engineering result measured in Appendix D.

Section 2 places the design among earlier market makers. Sections 3–5 work over the real numbers; Section 6 and Appendices B–C treat integer execution. Section 7 collects the assumptions and risks that the results leave open. Unless specified otherwise, amounts are measured in raw token units, reserves are nonnegative, and each transfer debits the sender and credits the recipient by the same amount. The paper covers the pool, its router and its liquidity token at contract commit `258021b`. It does not cover the managed-liquidity vault, cross-chain bridging or off-chain route selection.

2 RELATED WORK

Constant-function market makers. Uniswap v2 trades along $xy = k$ over the full price range, with fungible liquidity shares and fees that compound into reserves [3]. Angeris et al. analyse such markets and show that arbitrage keeps their prices close to external reference prices [4, 5]. Curve's StableSwap concentrates liquidity around a peg with a hybrid invariant [6].

Range positions. Uniswap v3 lets each provider choose tick-aligned bounds; within its range a position trades like a constant-product pool with virtual reserves [1]. The bin curve of Equation (3.2) is the v3 position curve for a range of one bin. The difference is that all providers in a bin share that range, so the bin can issue fungible shares and keep its fees in reserves. Elsts derives v3 liquidity from token amounts and shows that the price can be solved for as well [7]; a Zest CLMM bin stores no price, and Equation (3.5) recovers both from reserves.

Discretized liquidity. iZiSwap places liquidity at discrete price points and uses them for limit orders [8]. Liquidity Book partitions price into bins with fungible per-bin tokens; each bin is a constant-sum curve, so a swap within a bin executes at one price and only the active bin holds both tokens [2]. Meteora's DLMM applies the same design on Solana [9], and Bitflow's HODLMM brings bin-based liquidity to Stacks [11]. Liquidity Book and DLMM both charge a composition fee when a deposit changes the active bin's composition, approximating the swap fee on the amount exchanged internally. Maverick v2 is the closest design: each tick is a bounded constant-product curve, providers hold pro-rata claims on its reserves, and liquidity is computed from those reserves and the tick's bounds [10]. Zest CLMM shares that bin model. It differs in charging the exchange implied by adding and immediately redeeming liquidity (Section 5.3) and in its integer implementation with a proved one-sided error bound.

Market risk. Milionis et al. show that a market maker's providers lose, relative to a rebalancing portfolio, an amount they call loss-versus-rebalancing, set by price volatility and pool depth [12], and extend the analysis to pools with fees and discrete blocks [13]. Front-running and sandwich attacks on market-maker trades are documented in [14, 15]. Section 7 discusses both for this design.

3 DISCRETE LIQUIDITY

Table 1 lists the notation of Sections 3–5 and the contract fields that hold each quantity.

Table 1: Notation and contract fields. Rates are stored in units of 10^{-8}.

Symbol	Meaning	Contract field
X, Y	quote and base tokens	`token-x`, `token-y`
β	relative bin width	`bin-step` (bp)
t	bin index	`tick`
a, b	square-root price bounds	a, b (frame form)
x, y	real reserves	`balance-x`, `balance-y`
s, P	square-root price, price	derived
L	liquidity	derived
S	share supply	`total-supply`
f_X, f_Y	directional fee rates	`fee-rate-x`, `fee-rate-y`
ρ	LP fee fraction	`fee-rebate`
d	partner discount	`fee-discounts`

3.1 Price intervals

A pool exchanges quote token X for base token Y. Price is expressed as units of X per unit of Y. Each pool has a fixed relative bin width $\beta > 0$, defining the geometric grid

$$P_t = (1+\beta)^t, \qquad t \in \mathbb{Z}. \tag{3.1}$$

Bin t covers $[P_t, P_{t+1}]$. Write its square-root bounds as $a = \sqrt{P_t}$ and $b = \sqrt{P_{t+1}}$, with $0 < a < b$.

For liquidity $L > 0$, the bin's real reserves (x, y) satisfy

$$(x + La)\left(y + \frac{L}{b}\right) = L^2. \tag{3.2}$$

The virtual reserves are $U = x + La$ and $V = y + L/b$. With square-root price $s \in [a, b]$, the same curve is parameterized by

$$\begin{aligned} x &= L(s-a), \\ y &= L\left(\frac{1}{s} - \frac{1}{b}\right). \end{aligned} \tag{3.3}$$

The marginal price is $P = U/V = s^2 = -dx/dy$. At $s = a$ the bin contains only Y; at $s = b$ it contains only X. An empty bin has no defined market price. Figure 1 shows the curve. Uniswap v3 quotes the price of its first token in its second and writes the same curve as $(x + L/\sqrt{p_b})(y + L\sqrt{p_a}) = L^2$ [1]; Equation (3.2) is that curve with the roles of the two tokens exchanged.

3.2 Liquidity from reserves

Expanding Equation (3.2) gives

$$\begin{aligned} cL^2 - BL - xy &= 0, \\ c = 1 - \frac{a}{b}, \qquad B &= \frac{x}{b} + ay. \end{aligned} \tag{3.4}$$

For $(x, y) \neq (0, 0)$, the unique positive solution is

$$\mathcal{L}(x, y) = \frac{B + \sqrt{B^2 + 4cxy}}{2c}. \tag{3.5}$$

Set $\mathcal{L}(0, 0) = 0$. A nonempty reserve pair therefore determines both liquidity and price. Conversely, Equation (3.3) reconstructs the reserves from (L, s).

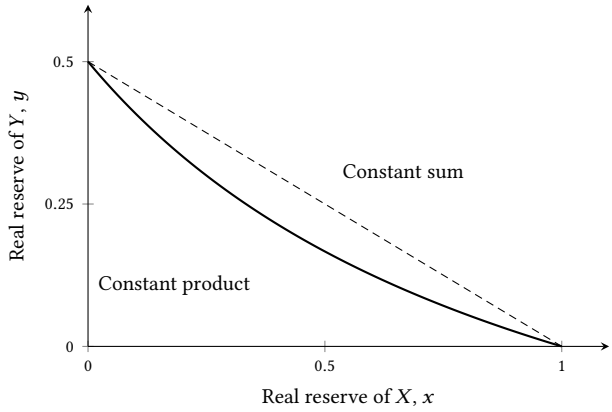


Figure 1: Real reserves of one bin with $\beta = 3$ ($a = 1, b = 2$) and $L = 1$. The solid curve is Equation (3.2); its marginal price $-dx/dy = s^2$ rises from $a^2 = 1$, where the bin holds only Y, to $b^2 = 4$, where it holds only X. The dashed line joins the same endpoints at the single price 2, as a constant-sum bin would.

The function $\mathcal{L}$ is increasing, concave and homogeneous of degree one. Appendix A derives these properties.

3.3 Capital concentration

At the geometric midpoint $s = \sqrt{ab}$, the position's value in X is

$$x + s^2 y = 2Ls\left(1 - \sqrt{a/b}\right). \tag{3.6}$$

A full-range position with invariant $xy = L_{\text{full}}^2$ has value $2L_{\text{full}}s$ at the same price. For equal capital,

$$\frac{L}{L_{\text{full}}} = \frac{1}{1 - (1+\beta)^{-1/4}} \sim \frac{4}{\beta} \quad (\beta \to 0). \tag{3.7}$$

This is a comparison of local liquidity at the midpoint. The concentrated position supplies two-sided liquidity over a narrower interval and becomes single-sided at its boundaries.

4 SWAPS

4.1 Execution within a bin

Consider an X-for-Y swap in a nonempty bin with accepted gross input $g \geq 0$, fee rate $0 \leq f < 1$ and net input $q = (1-f)g$. Along the curve with liquidity L, the output and new square-root price are

$$o = \frac{Vq}{U+q}, \tag{4.1}$$

$$s_1 = s_0 + \frac{q}{L}. \tag{4.2}$$

For a maximum price $P_{\max} > 0$, set $s_* = \min(b, \sqrt{P_{\max}})$. Given a gross input budget $G \geq 0$, the accepted net amount is

$$q = \min\left((1-f)G,\ L\left[s_* - s_0\right]_+\right). \tag{4.3}$$

Here $[z]_+ = \max(z, 0)$. The accepted gross amount is $g = q/(1-f)$. A zero amount produces no exchange. If execution reaches the upper edge, the output reserve is exhausted. For Y-for-X execution,

interchange the assets and use inverse price $Q = 1/P$ with square-root bounds $(1/b, 1/a)$. A lower limit $P_{\min} > 0$ on P becomes the upper limit $1/P_{\min}$ on Q.

For a nonzero trade, the average net price is $q/o = s_0 s_1$; the average gross price is $g/o = s_0 s_1/(1-f)$.

4.2 Fees

The pool specifies directional input fees $0 \le f_X, f_Y \le 1/10$ and an LP fee fraction $0 \le \rho \le 1$, common to its bins. The current parameters are held fixed throughout each swap, deposit or route. Use $f = f_X$ for X input and $f = f_Y$ for Y input; the same rates apply to deposit fees below. Following an X-for-Y exchange, the reserves become

$$(x', y') = (x + q + \rho f g,\ y - o). \tag{4.4}$$

The remaining $(1-\rho)fg$ is accounted for as protocol revenue. A positive LP fee credit increases liquidity per share without minting shares.

The price limit in Equation (4.3) applies to the net exchange. Adding the LP fee afterward can move the reserve-derived price beyond that limit, while it remains inside the bin. A marginal price limit alone does not constrain the fee rate, so every swap also takes a minimum output, a lower bound on the amount received.

A public pool may grant a transaction sender a discount $0 \le d \le 1$ on both rates. In the contract's integer units ($F = 10^8 f$, $D = 10^8 d$) the sender pays $F - \lfloor FD/10^8 \rfloor$: never more than F, and nothing when $D = 10^8$. The same rates apply to the sender's deposit fees (Section 5.3), and ρ applies to the discounted fee. Private pools (Section 5.4) take no discounts.

4.3 Routing

Pool liquidity is the collection of its independent bin inventories. An X-for-Y route traverses bins in ascending price order; the reverse route uses descending order. Each bin is filled at most once, subject to its boundary and the trader's price limit. Bins without output reserves are skipped. A route may permit a partial fill or require the entire input to be consumed, and an aggregate minimum output can constrain the settlement.

For fixed initial states on the exact grid and a common constant directional fee, price order maximizes output for a fixed input budget among single-direction allocations that fill each bin at most once. Marginal output per unit of gross X input is $(1-f)/P$; each bin's price interval precedes the next, with a possible tie at the shared boundary. The exchange argument uses that each bin's output is concave in its input and that bin intervals do not overlap. Rounded bin endpoints in the implementation can overlap slightly.

Each trade changes the inventories it touches; untouched bins keep their reserves. In a pool's normal state, bins above the price hold only Y at their lower edges and bins below it hold only X at their upper edges, so when a route empties one bin, the next starts at the price where the last one ended (Figure 2). The price path is continuous and bends where liquidity changes, as a Uniswap v3 price path does at an initialized tick. A bin out of line with its neighbours holds both tokens, and a route reaches it at a slightly worse price, never a better one, since every bin's price lies inside its interval and routes walk bins in price order. If, after fees, a lower bin sells Y below the price at which a higher bin buys it, an arbitrageur can buy Y from the lower bin and sell it to the higher one, pushing both bins toward their adjoining edges.

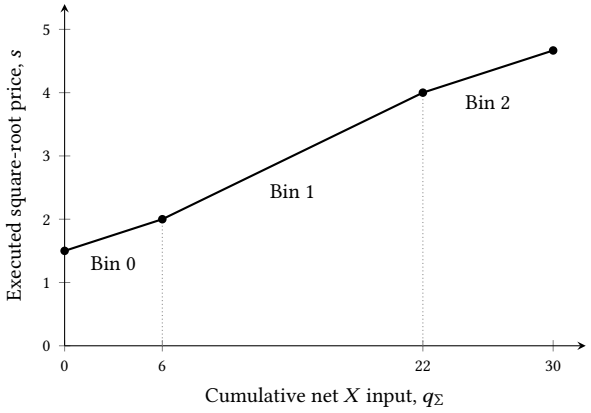


Figure 2: A fee-free route in a pool's normal state, with $\beta = 3$. Bin 0 starts two-sided at $s = 3/2$ with $L = 12$; bins 1 and 2 hold only Y at their lower edges, with $L = 8$ and $L = 12$. Net inputs $(6, 16, 8)$ exhaust Y in the first two bins, and the third stops at the input budget $q_\Sigma = 30$, two-sided. Each bin starts where the last one ended, so the price is continuous and bends where liquidity changes. Dotted lines mark the bin edges; the ordinate is the square-root price in the bin being executed.

5 LIQUIDITY ACCOUNTING

5.1 Shares and proportional changes

Let $S > 0$ denote a nonempty bin's outstanding share supply. A holder of $0 \le m \le S$ shares redeems the fraction m/S of both reserves:

$$(x_{\text{out}}, y_{\text{out}}) = \frac{m}{S}(x, y). \tag{5.1}$$

For a proportional deposit $(\Delta x, \Delta y) = \lambda(x, y)$ with $\lambda \ge 0$, issue λS shares. Both operations preserve L/S and price while the bin remains nonempty. In the model, a full redemption empties the bin and leaves neither quantity defined; the implementation prevents this by locking 1000 shares on a bin's first deposit (Section 6). The initial deposit establishes the bin's reserve composition and initial share scale; it need not match an external market price.

5.2 The exchange implicit in a deposit

For a general nonnegative deposit $d = (\Delta x, \Delta y)$ into a nonempty bin with reserves $r = (x, y)$ and share supply $S > 0$, write $L_0 = \mathcal{L}(r) > 0$ and $L_g = \mathcal{L}(r + d)$. Ignoring fees, issuing $S(L_g/L_0 - 1)$ shares and immediately redeeming those new shares leaves

$$r_{\text{remaining}} = \frac{L_0}{L_g}(r + d). \tag{5.2}$$

By homogeneity, its liquidity is L_0. The net reserve change

$$n = \frac{L_0}{L_g}(r + d) - r \tag{5.3}$$

is thus an exchange along the original curve. Since $\mathcal{L}$ is strictly increasing in each reserve, any nonzero n must have components of opposite sign. A proportional deposit gives $n = 0$; an imbalanced deposit can therefore exchange assets even when no explicit swap is called.

The positive component measures implicit input and cannot exceed the amount deposited in that token, since $L_0/L_g \le 1$. A deposit of the asset already held by a single-asset bin is proportional and has zero implicit input.

5.3 Deposit fees and issuance

Using the pool's directional fee rates, define for $i \in \{X, Y\}$

$$e_i = f_i \min\left(\Delta i, \frac{[n_i]_+}{1 - f_i}\right), \tag{5.4}$$

where $\Delta X = \Delta x$ and $\Delta Y = \Delta y$. The gross-up uses the same fee-on-gross-input convention as a swap, with the charge limited by the available deposit. A first deposit has no pre-existing position against which to exchange and incurs no conversion fee.

The LP fee share is credited to existing holders before new shares are issued. Let $e = (e_X, e_Y)$ and define

$$\begin{aligned} L_b &= \mathcal{L}(r + \rho e), \\ L_f &= \mathcal{L}\big(r + d - (1 - \rho)e\big). \end{aligned} \tag{5.5}$$

The new share amount is

$$m = S\left(\frac{L_f}{L_b} - 1\right). \tag{5.6}$$

Existing holders retain liquidity $SL_f/(S + m) = L_b \ge L_0$. Since $e \le d$ componentwise, $L_f \ge L_b$ and $m \ge 0$. Protocol revenue is excluded from reserves backing shares. Fees are assessed separately for each deposit using current bin reserves. The rule uses the fee-free exchange in Equation (5.3) as its conversion measure; it does not assert equivalence to a separately executed swap followed by a proportional deposit. The rule is designed so that depositing and immediately withdrawing cannot beat a swap at the same fee rate. Property fuzzing of the contract has found no round trip that does, but this paper does not prove it.

Example. Take a bin with $a = 1$ and $b = 1.1$ (so $\beta = 0.21$), liquidity $L_0 = 1000$ and $s = 1.05$. Its price is $P = 1.1025$, its reserves are $r \approx (50,\ 43.290)$, and let $S = 1000$. A deposit $d = (100, 0)$ of X alone gives $L_g \approx 2011.70$, and Equation (5.3) gives $n \approx (24.564,\ -21.771)$: the deposit implicitly sells about 24.56 units of X for 21.77 units of Y. With $f_X = 1\%$ and $\rho = 1/2$, Equation (5.4) charges $e_X = f_X n_X/(1 - f_X) \approx 0.2481$, the fee on a swap with net input n_X. Then $L_b \approx 1001.27$, $L_f \approx 2010.45$, and Equation (5.6) issues $m \approx 1007.90$ shares, against 1011.70 without the fee. Without the fee, depositing and immediately redeeming would perform the same exchange at no cost.

5.4 Private pools

A pool may name a single depositor. Only the depositor's deposits mint shares; swaps and redemptions are unrestricted, so a former depositor or any other holder can always redeem. Private pools are not indexed by token pair and bin step, so several of them and a public pool can share one. The bin mathematics of Sections 3 to 5 is unchanged. Governance can move a private pool to a new depositor but cannot make a public pool private or the reverse.

6 IMPLEMENTATION ON STACKS

Clarity provides native 128-bit integers and bounded sequence types [16]. Direct evaluation of the bin equations can exceed that word size even when the final amount is representable. Zest CLMM combines binary coordinate changes, directed approximation and multiword division to execute the model within a finite numerical domain.

6.1 Normalized price coordinates

Tick square-root prices are approximated by $\mu 2^{\varepsilon}$, where $1 \le \mu < 2$ is a fixed-point mantissa with 60 fractional bits. Keeping the exponent separate preserves relative precision across widely different raw token prices. For $E = 2\varepsilon$, the bin works in a rescaled coordinate system, its *frame*, with reserves

$$u = x2^{\max(0,-E)}, \qquad v = y2^{\max(0,E)}. \tag{6.1}$$

These shifts are exact within the admitted amount range. The represented bin retains the constant-product form with local square-root bounds near one. Geometry is computed when a bin is initialized and cached with its reserves and share supply; liquidity and price remain reserve-derived.

The frame has a fixed reserve cap. Consequently, its capacity in raw token units depends on the exponent. Bin widths, tick ranges and reserve caps must be certified together. The ideal geometric grid and its represented endpoints are distinct; the coordinate change is exact for the latter.

6.2 Conservative liquidity evaluation

For a nonempty bin, a second, common scaling of both frame reserves uses the homogeneity of $\mathcal{L}$. The larger normalized reserve lies in $[2^{61}, 2^{62})$, allowing the quadratic to be evaluated with native multiplication and integer square root. Right shifts in this normalization discard low bits and are included in the error bound. Directed approximations to $c = 1 - a/b$ are derived from the represented endpoints.

The result $\widetilde{L}$ underestimates the represented curve's exact liquidity L^*. Appendix C derives an interval

$$\widetilde{L} \le L^* \le \widetilde{L} + \ell(\widetilde{L}). \tag{6.2}$$

To keep every rounding error in the pool's favour, the implementation uses the upper liquidity estimate for the reserve a trader pays into and the lower estimate for the reserve paid out. Trader price limits are rounded toward the current price; limits too large to represent are replaced by a value above the bin, so they never bind. The interval supplies the virtual-reserve and share bounds in Appendix B. The implementation applies them in frame units

and deducts a further 2^{-40} of each output plus one frame unit per step. The LP fee credit is rounded down, so the protocol's share rounds up. The input-to-target cap rounds down at a trader's limit and up only at a bin edge, where nonnegative reserves ensure confinement. Rounding can leave residual balances; a step producing no output accepts no input. Deposits require an enclosure of the post-deposit price to lie within the supplied limits. Rounding uncertainty can cause narrow but valid price windows to be rejected.

6.3 Exact multiword arithmetic

Fee calculations, proportional payouts and swap ratios require products that can be wider than their results. For unsigned 128-bit operands and a positive divisor, the multiply–divide primitive returns the exact floor or ceiling of the quotient when that result fits in 128 bits. Otherwise it returns an error; a zero divisor also returns an error.

Products that fit use native arithmetic. Larger products are formed from 64-bit limbs and divided by an adaptation of Knuth's Algorithm D [19]. Normalization bounds the quotient-digit correction, replacing a bit-by-bit division loop with a fixed number of limb operations. Appendix C gives the quotient and overflow conditions.

6.4 Bounded routes and settlement

A pool route takes a caller-selected list of at most 32 strictly ordered ticks. The pool walks that list internally, applying the single-bin swap calculation and carrying route parameters and running totals in a fixed-size accumulator. Pool and token checks run once. A successful route with nonzero output transfers the aggregate input and output once each and records protocol fees once. Bin reads depend on the bins examined while input remains, and bin writes on those traded.

A batched withdrawal takes up to 64 pairs of tick and fraction, applies the redemption of Equation (5.1) to each bin in turn, skips a bin whose share burn rounds to zero, and transfers each token once. Each bin ends as it would after its own redemption. The first deposit into a bin mints 1000 shares to the pool that can never be redeemed, so a bin's share supply never returns to zero.

The router chains up to nine pools in one transaction. Each hop must fill completely and carries no price limit or minimum output of its own; only the route's final minimum output and deadline protect the trader. The router also batches up to 100 deposits into one pool.

Full-fill and minimum-output checks apply to the accumulated result. Propagated errors abort the public call and roll back its state changes and transfers [17]. Aggregation preserves the bin calculations for the same ordered list, initial states and fee parameters. Appendix D measures this change against a per-bin execution reference.

Every token call, including balance reads, runs inside a Clarity asset allowance that bounds the assets it may move [18]. Incoming transfers must report the expected increase in the pool's balance, and outgoing transfers the expected decrease in the pool's balance and increase in the recipient's. These checks retain the model's assumption of standard token transfer semantics.

7 RISKS AND LIMITATIONS

The results above hold under stated assumptions. This section collects them, together with the risks they leave to providers, traders and the protocol.

7.1 Liquidity providers

Shares are claims on current reserves, not on the amounts deposited. A bin traded through its range ends holding only the asset that has fallen in relative value, and the provider bears the difference from simply holding the deposit; fees may or may not offset it. A position does not rebalance itself as the market moves. Bins are independent: liquidity away from the market price behaves like a resting limit order and is filled when the external price crosses it, so arbitrageurs trade against stale bin prices. Milionis et al. quantify this adverse-selection cost as loss-versus-rebalancing [12]. This paper makes no claim that fees cover it. The valuation bound of Appendix A applies to one bin, under swaps alone, at its initial price.

A proportional deposit pays no fee, so liquidity can be added just before a large swap and removed after it, diluting the fees of providers who stay. A bin's first depositor sets its share scale; with ρ near one, fee credits can raise the value per share far enough that later deposits lose a noticeable amount to share rounding, so depositors should set a minimum share amount from the value they deposit.

7.2 Traders

Pending transactions are public before execution. On Stacks, the miner of each tenure orders transactions from a public mempool, so a trade can be preceded by others, as in a sandwich attack [14, 15]. Every swap takes a price limit, a minimum output and a deadline in seconds, checked against the block's timestamp; these bound the terms a trader accepts. Deposits take price limits and a minimum share amount but no deadline. A multi-pool router path has no per-hop price limit; only its final minimum output protects the trader. Fee rates are capped at 10%, and a price limit does not bound the fee, so traders should set the minimum output tightly. The pool settles with the transaction sender's tokens, including when it is reached through another contract, so wallets should submit transactions with post-conditions in deny mode, which abort on any token movement they do not list.

A private pool has a single depositor. Its traders have the same protection as in a public pool: a price limit, a minimum output and a deadline. Private pools are not indexed, so they are found through events.

7.3 Tokens

The model assumes standard transfer semantics, and only standard SIP-010 tokens [20] should be listed. Transfers are checked against balance changes and every token call runs inside an asset allowance (Section 6), so a token that delivers less than requested makes the call revert. Withdrawals paying that token then revert, and a two-token withdrawal also holds back the other asset. The same holds for a token whose issuer can pause transfers or blacklist the pool, and an upgradeable token can change its behaviour

after listing. Tokens whose balances change outside transfers, such as rebasing tokens, would break reserve accounting and must not be listed. Listing is a governance decision, and each pool's safety depends on it.

7.4 Numerical limits

Integer execution is conservative, not exact (Appendix B). Traders receive slightly less than the exact curve output. A deposit forgoes up to about 2^{-54} of the bin's value to the liquidity error margin (Appendix B), which existing providers keep, so a very small deposit into a large bin can mint no shares and is rejected. Rounding leaves residual balances at bin edges and can reject narrow but valid price windows. The LP fee credit can move a bin's resting price slightly past a trader's limit, although the trader's own execution respects it (Section 4). The arithmetic bounds of Appendix C are certified for bin widths of 1%, 5%, 10% and 20% only, and the contract accepts no other width. The examples in this paper use wider bins for legibility.

7.5 Governance

Pools are administered by an ExecutorDAO. Proposals are approved by an operators multisig and execute as soon as they reach its threshold; there is no timelock, and a proposal that does not reach the threshold within 241,920 blocks expires. The operators and the threshold can be changed only by proposal; on Stacks mainnet, any two of three operators can approve one. Any extension a proposal enables holds every administrative power directly; at bootstrap the only extension is the operators multisig.

Governance can list public and private pools; set fee rates up to 10% and the LP fee fraction; pause swaps and deposits, or close a pool to deposits; claim protocol fees; register native-STX token wrappers before a token is listed; move a private pool to a new depositor; freeze transfers of liquidity shares; set the share token's metadata URI; and grant partner discounts on public pools.

The pool contract exposes no governance function that mints, burns or redeems liquidity shares, withdraws reserves or blocks withdrawals, which remain available in paused pools and while share transfers are frozen. The contracts are not upgradeable; a new version requires providers to migrate voluntarily.

7.6 Review status

The contracts were reviewed internally, and the pool, router, liquidity token and governance contracts were also audited externally by a third-party security provider. The fixes for its findings are in contract commit `258021b`, which this paper describes, and the contracts deployed on Stacks mainnet match that commit's code. The internal review is available on request [21]. Supporting lemmas for the arithmetic are proved with the Z3 solver; the liquidity error bound of Appendix C is proved by hand with a computer-assisted constant.

8 CONCLUSION

Zest CLMM gives each geometric bin its own bounded constant-product curve and fungible shares. Liquidity and price follow from reserves, retained fees compound into them, and a deposit that changes a bin's composition pays the swap fee on the exchange it implies. The Clarity implementation evaluates the model with exact multiword arithmetic and liquidity estimates that err only downward, and settles multi-bin routes with one transfer per token. Questions this paper leaves open include the choice of bin width for a given pair, fees that respond to volatility, and how bin portfolios perform against loss-versus-rebalancing in practice.

A PROPERTIES OF RESERVE-DERIVED LIQUIDITY

Let $D = \sqrt{B^2 + 4cxy}$ for fixed $0 < a < b$. Differentiating Equation (3.4) at a nonempty state gives

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{V}{D}, \qquad \frac{\partial \mathcal{L}}{\partial y} = \frac{U}{D}. \tag{A.1}$$

Both are positive, with $(\partial\mathcal{L}/\partial x)/(\partial\mathcal{L}/\partial y) = 1/P$. Homogeneity follows directly from Equation (3.5). A second differentiation yields

$$\nabla^2 \mathcal{L} = -\frac{2}{D^3}\begin{pmatrix} y^2 & -xy \\ -xy & x^2 \end{pmatrix}. \tag{A.2}$$

For any direction $v = (v_x, v_y)$, $v^\top \nabla^2 \mathcal{L}\, v = -2(yv_x - xv_y)^2/D^3 \le 0$. Thus $\mathcal{L}$ is concave and linear along reserve rays. Along a segment between non-collinear reserve vectors, $yv_x - xv_y$ is constant and nonzero, so concavity is strict. For nonzero r_1, r_2, homogeneity and concavity imply

$$\mathcal{L}(r_1 + r_2) \ge \mathcal{L}(r_1) + \mathcal{L}(r_2), \tag{A.3}$$

with equality exactly when the reserve vectors are proportional. This explains why combining differently composed deposits changes liquidity beyond their separate contributions.

Proposition A.1. *Consider a sequence of swaps in one bin with fixed endpoints, nonnegative retained fees, nonnegative reserves, and output no greater than the smaller of the output reserve and the exact curve output for the same accepted net input from the pre-swap state. Exclude liquidity additions, withdrawals and external changes to token balances. If $r_0 = (x_0, y_0)$ is the initial nonempty state and P_0 its price, then for the final reserves $r_1 = (x_1, y_1)$,*

$$x_1 + P_0 y_1 \ge x_0 + P_0 y_0. \tag{A.4}$$

Proof. Each swap weakly increases $\mathcal{L}$, hence $\mathcal{L}(r_1) \ge \mathcal{L}(r_0)$. Concavity gives

$$\mathcal{L}(r_1) - \mathcal{L}(r_0) \le \nabla\mathcal{L}(r_0) \cdot (r_1 - r_0).$$

By Equation (A.1), the gradient is a positive multiple of $(1, P_0)$, proving Equation (A.4). □

The valuation is of one bin's reserves at its initial price; protocol revenue is accounted for separately.

B CONSERVATIVE INTEGER BOUNDS

Integer execution rounds swap outputs and newly issued shares downward from conservative estimates. The following bounds give sufficient conditions; their validity depends on the chosen price representation and arithmetic ranges. Rounding can dominate small transactions and leave residual balances at bin boundaries.

B.1 Conservative execution

Fix represented endpoints $0 < a < b$ and nonnegative integer reserves $(x, y) \neq (0, 0)$. Assume bounds $0 \le L_- \le \mathcal{L}(x, y) \le L_+$, relative to the represented curve, whose endpoints may approximate the ideal grid. For X-for-Y execution, form

$$\begin{aligned} U_+ &= x + \lceil L_+ a \rceil, \\ V_- &= y + \lfloor L_- / b \rfloor. \end{aligned} \tag{B.1}$$

Then $U_+ \ge U > 0$ and $0 \le V_- \le V$, so for integer net input $q \ge 0$,

$$\left\lfloor \frac{V_- q}{U_+ + q} \right\rfloor \le \frac{V q}{U + q}. \tag{B.2}$$

Capping the output at the available reserve preserves this inequality. Further deductions must leave nonnegative output. For an inward square-root limit $\bar{s} \in [a, b]$ with $\bar{s}^2 \le P_{\max}$, a sufficient input cap is

$$0 \le q \le \left[\lfloor L_- (\bar{s} - a) - x \rfloor \right]_+ . \tag{B.3}$$

For $q > 0$, this implies $s_0 + q/L \le \bar{s}$, since $s_0 = a + x/L$. Reducing output only lowers the resulting price before LP-fee credit. If $P_{\max} < a^2$, no input is accepted. The reverse direction uses inverse prices.

For the same net input, these bounds remove no more output than the exact curve permits. With valid liquidity bounds and arithmetic ranges, the represented curve's exact liquidity is nondecreasing under these swaps and nonnegative retained fees.

B.2 Share rounding

For a deposit into a nonempty bin with integer $S > 0$, take bounds $0 \le L_f^- \le L_f$ and $0 < L_b \le L_b^+$ for the quantities in Section 5. Issue

$$m = \left\lfloor S \frac{L_f^- - L_b^+}{L_b^+} \right\rfloor. \tag{B.4}$$

Accept only deposits producing $m > 0$. Then $L_f/(S + m) \ge L_b/S$. Withdrawals round each proportional payout downward. Redeeming every share would pay all integer reserves, but the 1000 shares locked on a bin's first deposit are never redeemed.

B.3 Deposit price limits

Use the post-deposit reserves after fee allocation. In the normalized frame of Appendix C, write $Q = 2^{60}$, $A = Q\alpha$, $B = Q\gamma$ and $0 < L_- \le \Lambda(u, v) \le L_+$. For $u, v > 0$, define integer square-root bounds

$$\begin{aligned} S_- &= A + \left\lfloor \frac{uQ}{L_+} \right\rfloor, \\ S_+ &= \min\left(B, A + \left\lceil \frac{uQ}{L_-} \right\rceil \right). \end{aligned} \tag{B.5}$$

The exact frame square-root price is $s^* = \alpha + u/\Lambda(u, v)$. Division by liquidity is decreasing, so $S_- \le Q s^* \le S_+$. Single-sided states use the exact endpoint A when $u = 0$ and B when $v = 0$. If the lower liquidity bound is zero, the full interval $[A, B]$ remains valid for a nonempty bin.

Accept a deposit only when the enclosure lies within the inward-rounded square-root limits. Limits above the conversion range use the above-bin sentinel described in Section 6. This certifies the resulting represented price against both limits. A rounded display price alone need not bound that price in either direction.

C FINITE-WORD ARITHMETIC

C.1 Coordinate change and domain

Let $Q = 2^{60}$ and write the represented square-root bounds as $a_r = \alpha 2^{\varepsilon}$, $b_r = \gamma 2^{\varepsilon}$, where $\varepsilon \in \mathbb{Z}$, $\alpha = A/Q$ and $\gamma = B/Q$. With $s_x = \max(0, -2\varepsilon)$, $s_y = \max(0, 2\varepsilon)$ and $L_f = 2^{|\varepsilon|} L_r$, define $u = 2^{s_x} x$, $v = 2^{s_y} y$. The identities $s_x + \varepsilon = s_y - \varepsilon = |\varepsilon|$ give

$$\begin{aligned} (u + \alpha L_f)(v + L_f/\gamma) &= 2^{s_x + s_y} (x + a_r L_r)(y + L_r/b_r) \\ &= L_f^2. \end{aligned}$$

Thus the frame change is exact for represented endpoints. Require integer shifts $0 \le s_x, s_y < 118$ and nonnegative integer raw amounts x, y below $2^{118 - s_x}$ and $2^{118 - s_y}$, respectively. Both shifted reserves then lie below $H = 2^{118}$, without truncation or overflow.

The following sufficient domain covers the four bin widths 1%, 5%, 10%, 20%:

$$\begin{gathered} 1 \le \alpha < 2, \quad \alpha < \gamma \le 11/5, \quad \gamma - \alpha \le 1/5, \\ 1/202 \le c = 1 - \alpha/\gamma \le 1/10. \end{gathered}$$

For these widths the integer step multipliers R satisfy $R < (11/10)Q$ and $201R - 202Q \ge 201$. With $Q \le A < 2Q$ and $B = \lfloor AR/Q \rfloor$, these imply the stated domain; in particular, $201B \ge 202A$. Other widths require new bounds.

Write $\Lambda(u, v)$ for exact liquidity in this frame and set $\Lambda(0, 0) = 0$. For a nonempty state,

$$T = u/\gamma + \alpha v, \qquad \Lambda = \frac{T + \sqrt{T^2 + 4cuv}}{2c}.$$

The derivatives in Appendix A give $\Lambda_u \le 1/(\alpha c)$ and $\Lambda_v \le \gamma/c$: each derivative is homogeneous of degree zero, so it depends only on u/v, is monotone in it, and is largest at the opposite reserve axis. Also $\Lambda \ge \max(u, v)/(\gamma - \alpha)$, since the quadratic is nonpositive at both $u/(\gamma - \alpha)$ and $\alpha\gamma v/(\gamma - \alpha)$.

C.2 Directed solve and error bound

For $m = \max(u, v) > 0$, let $e = \lfloor \log_2 m \rfloor$. If $e \ge 61$, set $k = e - 61$ and $(U, V) = (\lfloor u/2^k \rfloor, \lfloor v/2^k \rfloor)$. Otherwise set $d = 61 - e$ and $(U, V) = (2^d u, 2^d v)$. In both cases $2Q \le \max(U, V) < 4Q$.

With $h = 2^{70}$, the integer solve is

$$\begin{aligned} C &= \lfloor hc \rfloor + 1, & D_4 &= 4\lfloor Qc \rfloor, \\ \widehat{T} &= \lfloor U/\gamma \rfloor + \lfloor \alpha V \rfloor, & Z &= \lfloor UV/Q \rfloor, \\ \widehat{D} &= \widehat{T}^2 + D_4 Z, & \widehat{S} &= \lfloor \sqrt{\widehat{D}} \rfloor, \\ q &= \left\lfloor \frac{(\widehat{T} + \widehat{S}) 2^{63}}{C} \right\rfloor. \end{aligned}$$

Put $c' = C/h$, so $c < c' \le c + 1/h$. Numerator terms round downward, hence

$$64q \le \frac{\widehat{T}+\widehat{S}}{2c'} \le \Lambda(U,V).$$

The returned estimate is

$$\widetilde{L} = \begin{cases} 2^{k+6}q, & e \ge 61, \\ \lfloor 2^{6-d}q \rfloor, & e < 61. \end{cases}$$

Monotonicity and homogeneity preserve the lower bound $\widetilde{L} \le L^* = \Lambda(u,v)$.

For normalized reserves, write $T = U/\gamma + \alpha V$, $D = T^2 + 4cUV$, $N = T + \sqrt{D}$ and $\widehat{N} = \widehat{T} + \widehat{S}$. The rounding losses satisfy

$$\begin{aligned} 0 &\le T - \widehat{T} < 2, \\ 0 &\le D - \widehat{D} < 4T + 4UV/Q + 4cQ, \\ 0 &\le N - \widehat{N} < 7 + 4UV/(QT) + 4cQ/T. \end{aligned}$$

The last line uses $\sqrt{D} + \sqrt{\widehat{D}} \ge T$ and a square-root rounding loss below one. Since $T \ge 10Q/11$, $N \ge 2T$ and $UV/T^2 \le 1/[4(1-c)] \le 5/18$,

$$\frac{N-\widehat{N}}{N} < \frac{77/20 + 5/9 + 121/500}{Q} < \frac{5}{Q}.$$

Rounding c upward contributes at most $1/(hc) \le 1/(4Q)$ in relative error. Thus the solve before quantization loses at most $21\Lambda(U,V)/(4Q)$. Quantizing to $64q$ loses less than 64, whose relative contribution is at most $32/(5Q)$ because $\Lambda(U,V) \ge 2Q/(\gamma-\alpha)$.

When reserves are downscaled, each coordinate loses less than one. The derivative bounds give a relative liquidity loss at most

$$\frac{\gamma/\alpha + \gamma^2}{\max(U,V)} \le \frac{88}{25Q}.$$

Upscaling loses no reserve precision. Restoring the scale preserves these relative bounds; any final right shift loses less than one original liquidity unit. Therefore

$$\begin{aligned} 0 \le L^* - \widetilde{L} &\le \frac{1517}{100Q} L^* + 1 \\ &\le 2^{-56} L^* + 1. \end{aligned}$$

The coefficient is $21/4 + 32/5 + 88/25 = 1517/100 < 16$. The zero state is exact.

For $\delta = 2^{-56}$, the error bound implies $L^* \le (\widetilde{L}+1)/(1-\delta)$. For $t \ge 2$, $(1-2\delta)t \ge 1$, so

$$t + \lfloor 2\delta t \rfloor + 2 > t + 2\delta t + 1 \ge \frac{t+1}{1-\delta}.$$

The cases $t = 0, 1$ follow directly. Consequently the integer interval used in Section 6 is

$$\begin{aligned} \widetilde{L} \le L^* &\le \widetilde{L} + \ell(\widetilde{L}), \\ \ell(t) &= \lfloor t/2^{55} \rfloor + 2. \end{aligned}$$

C.3 Word-size bounds

All native solve operations fit in 128 bits. Indeed, $U, V < 4Q$, $\widehat{T} < 12Q$, $Z < 16Q$ and $D_4 < 4Q$ imply

$$\begin{aligned} \widehat{D} &< 208Q^2 < 2^{128}, \\ \widehat{S} &< 15Q, \\ (\widehat{T}+\widehat{S})2^{63} &< 216Q^2 < 2^{128}. \end{aligned}$$

The component products UQ, AV and UV are below $4Q^2$, $8Q^2$ and $16Q^2$, respectively. Frame construction also fits: $AR < (11/5)Q^2$ and $(B-A)2^{70} \le Q2^{70}/5 < 2^{128}$. All divisors are positive.

For outer rescaling, let $\lambda = \Lambda(1,1)$ and $T_1 = \alpha + 1/\gamma$. Using $\sqrt{T_1^2 + 4c} \le T_1 + 2c/T_1$ gives

$$\begin{aligned} \gamma\lambda &\le (\alpha\gamma + 1)/c + \gamma/T_1 \\ &\le \frac{5}{c(1-c)} \le K, \qquad K = \frac{204020}{201} < 1016. \end{aligned}$$

The second inequality uses $1-c = \alpha/\gamma$ and $\alpha < 2$: then $(1+\alpha\gamma)/c < (5-c)/(c(1-c))$ and $\gamma/T_1 = 1/\big((1-c) + 1/\gamma^2\big) < 1/(1-c)$.

Monotonicity gives $L^* \le H\lambda$. Since $\gamma \ge 1$, this also bounds the returned left shifts below 2^{128}. Writing $L_+ = \widetilde{L} + \ell(\widetilde{L})$, both $u + \lceil \alpha L_+ \rceil$ and $v + \lceil L_+/\gamma \rceil$ are below

$$[1 + (1 + 2^{-55})K]H + 5 < 1024H = 2^{128}.$$

The same bound applies to γL_+. Swap denominators remain in range when the accepted input leaves the corresponding real frame reserve below H.

The deposit price bounds also fit. For integer $m = \max(u,v) \ge 1$, $L^* \ge 5m$ and the error bound give $\widetilde{L} > 4m - 1$, so the integer $\widetilde{L}$ satisfies $\widetilde{L} \ge 4m$. Consequently $uQ/\widetilde{L} \le Q/4$ and $A + \lceil uQ/\widetilde{L} \rceil < 9Q/4 + 1 < 2^{62}$, before capping at B.

C.4 Exact multiply–divide

Let $W = 2^{128}$ and $R = 2^{64}$. For operands $0 \le a, b < W$ and $0 < d < W$, form the exact product $ab = p_h W + p_l$ from four 64-bit limb products, with $0 \le p_l < W$. Then

$$\left\lfloor \frac{ab}{d} \right\rfloor < W \quad \Longleftrightarrow \quad p_h < d.$$

Necessity follows from $ab \ge p_h W$; sufficiency from $ab < (p_h + 1)W \le dW$.

After rejecting $p_h \ge d$, two native divisions suffice when $d < R$. Otherwise set $s = 127 - \lfloor \log_2 d \rfloor$ and normalize the divisor to $D = d2^s$, whose high limb $d_1 \ge R/2$. The initial partial remainder satisfies

$$r_0 = p_h 2^s + \lfloor p_l 2^s / W \rfloor < D,$$

since $p_h \le d-1$ and the carry is below 2^s. For partial remainder $r < D$ and next limb $x < R$, the quotient digit is $q = \lfloor (rR+x)/D \rfloor < R$. Knuth's normalized estimate [19]

Table 2: Swap-transaction costs; setup and verification queries excluded.

Cost dimension	Sequential	Pool walk
One bin		
Runtime units	402,625	380,371
Reads	29	29
Writes	6	6
Bytes read	63,428	66,165
Bytes written	313	313
Ten bins		
Runtime units	3,787,678	2,043,994
Reads	263	47
Writes	60	15
Bytes read	597,695	68,334
Bytes written	3,130	2,482

$$\widehat{q} = \min(\lfloor r/d_1 \rfloor, R-1) \quad \text{satisfies} \quad q \le \widehat{q} \le q+2.$$

To see the correction bound, put $z = \lfloor r/d_1 \rfloor$ and $D = d_1R + d_0$. If $q > z$, then $qd_1 \ge r+1$, contradicting $qD \le rR + x$. Also

$$(z-q-1)d_1R < (q+1)d_0 < R^2,$$

so $z-q \le 2$; capping z preserves the bounds. Exact limb comparisons select the greatest nonnegative candidate among $\widehat{q}, \widehat{q}-1, \widehat{q}-2$ whose product with D does not exceed $rR+x$. A decrement is evaluated only after the preceding candidate fails, so it cannot underflow. Two digits from the normalized low word yield the floor quotient. The normalized remainder is 2^s times the original remainder; its zero test therefore determines whether the ceiling adds one, with overflow rejected. These are applications of Algorithm D, rather than approximations to division.

D ROUTE EVALUATION

We compare a pool-internal walk with a sequential router calling the same single-bin entry point once per tick. Both use the pool arithmetic at commit `f600bd1` [22], executed with Clarity 5, epoch 3.4 and Clarinet SDK/WASM 3.24.1. Each case starts from a fresh simnet state using standard mock tokens.

Table 2 reports X-for-Y routes over one or ten bins of width 1%, starting at tick zero. Each bin contains 10^{12} raw units of Y and no X. The gross input budget is 10^{14}, the fee rate is 1%, and half the fee remains with LPs. Partial fills are allowed, the minimum output is zero and the price limit does not bind. All supplied bins trade.

Two runs produced identical measurements. For these workloads, both directions matched the sequential reference in route response, bin reserves, share supply, protocol fees and token balances. Aggregate settlement reduces ten-bin token transfers from twenty to two. One-bin bytes read increase because the aggregate router's source is larger. These cases measure execution costs, not transaction fees, throughput or worst-case capacity. The measured code predates the audit fixes in commit `258021b`, so current costs may differ slightly. The reproduction materials, available on request [22], contain the probes, complete results and pinned environment.

The measurements predate the partner discounts, private pools, batched withdrawal, payout delivery checks, deposit price-bound fix and audit fixes (swap deadlines and token calls inside the asset allowance) described in Sections 4–7, and have not been repeated since. Those changes leave the bin mathematics, rounding and fee split unchanged but add work to every swap, so these figures understate the current costs.

REFERENCES

[1] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. 2021. *Uniswap v3 Core*. https://app.uniswap.org/whitepaper-v3.pdf.

[2] MountainFarmer, Louis, Hanzo, Wawa, Murloc, and Fish. 2022. *JOE v2.1 Liquidity Book*. Trader Joe. https://github.com/lfj-gg/LB-Whitepaper.

[3] Hayden Adams, Noah Zinsmeister, and Dan Robinson. 2020. *Uniswap v2 Core*. https://app.uniswap.org/whitepaper.pdf.

[4] Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra. 2021. An Analysis of Uniswap Markets. *Cryptoeconomic Systems*. arXiv:1911.03380.

[5] Guillermo Angeris and Tarun Chitra. 2020. Improved Price Oracles: Constant Function Market Makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT '20)*. arXiv:2003.10001.

[6] Michael Egorov. 2019. *StableSwap: Efficient Mechanism for Stablecoin Liquidity*. https://www.curve.finance/files/stableswap-paper.pdf.

[7] Atis Elsts. 30 September 2021. *Liquidity Math in Uniswap v3*. Technical note. https://atiselsts.github.io/pdfs/uniswap-v3-liquidity-math.pdf.

[8] J. Yin and M. Ren. 2021. *iZiSwap: Building Decentralized Exchange with Discretized Concentrated Liquidity and Limit Order*. https://github.com/izumiFinance/iZiSwap-core/tree/main/whitepaper.

[9] Meteora. *What is DLMM*. Technical documentation. https://docs.meteora.ag/core-products/dlmm/what-is-dlmm. Accessed October 2026.

[10] Maverick Protocol. *Maverick v2*. Technical documentation, https://docs.mav.xyz, and `TickMath.getTickL`, https://github.com/maverickprotocol/v2-common. Accessed October 2026.

[11] Bitflow. *HODLMM API Documentation*. https://docs.bitflow.finance/bitflow-documentation/developers/hodlmm-api-documentation. Accessed October 2026.

[12] Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. 2022. *Automated Market Making and Loss-Versus-Rebalancing*. arXiv:2208.06046.

[13] Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. 2023. *Automated Market Making and Arbitrage Profits in the Presence of Fees*. arXiv:2305.14604.

[14] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *2020 IEEE Symposium on Security and Privacy*.

[15] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V. Le, and Arthur Gervais. 2021. High-Frequency Trading on Decentralized On-Chain Exchanges. In *2021 IEEE Symposium on Security and Privacy*.

[16] Stacks. *Clarity Type Reference*. https://docs.stacks.co/reference/clarity/types. Accessed October 2026.

[17] Stacks. *Clarity Function Reference*. https://docs.stacks.co/reference/clarity/functions. Accessed October 2026.

[18] Stacks Improvement Proposal 033. *Clarity 4*. https://github.com/stacksgov/sips/blob/main/sips/sip-033/sip-033-clarity4.md.

[19] Donald E. Knuth. 1997. *The Art of Computer Programming*, Vol. 2: *Seminumerical Algorithms*, 3rd ed., §4.3.1. Addison-Wesley.

[20] Stacks Improvement Proposal 010. *Standard Trait Definition for Fungible Tokens*. https://github.com/stacksgov/sips/blob/main/sips/sip-010/sip-010-fungible-token-standard.md.

[21] Zest Protocol. 2026. *Zest CLMM internal review*. Available on request.

[22] Zest Protocol. *CLMM reference implementation*, commit `f600bd1`, with benchmark probes. Available on request.

DISCLAIMER

This paper is for general information only. It is not investment advice, a recommendation or solicitation to buy or sell any asset, or accounting, legal or tax advice. It describes software that is proprietary to Zest Protocol, whose contributors have a financial interest

in its use. The paper reflects the design at the time of writing; the contracts and the paper may change without notice, and the paper may not be updated to reflect those changes. It is provided as is, without warranty of any kind, and is not an offer of any token or security. Statements about future behaviour or plans are forward-looking and may not come true. Smart contracts can contain errors, and using them can result in the loss of funds.