CenturionDEX v3: Worked Examples with Real Numbers

Concentrated Liquidity, Swaps, Fees, Oracles, and Position Management

Centurion Labs*

Abstract

This note presents a self-contained collection of fully worked numerical examples for the core operations of a CenturionDEX v3 pool: tick math, minting and burning concentrated liquidity positions, single- and multi-interval swaps, the position-value function, impermanent loss, multi-position aggregation, fee accounting via outside/inside fee growth, price impact and slippage, range (limit) orders, the time-weighted average price (TWAP) oracle, protocol fees, flash swaps, rebalancing, and a capital-efficiency comparison against a full-range constant-product pool. All examples use token X = CTN (the native CenturionDEX asset) and token Y = USDC, with the convention that the market price p is expressed as Y per unit of X, i.e. USDC per CTN. Prices are chosen so that all $\sqrt{p}$ values are clean rationals, which makes every reserve, swap amount, and fee computation exactly reproducible by hand. A short appendix collects the underlying identities and a fixed-point arithmetic note on the $Q64.96$ representation used on-chain.

Reader's guide

The presentation is self-contained but technical. Readers unfamiliar with concentrated-liquidity automated market makers (AMMs) may prefer to read this guide first; each subsequent section also opens with a short pedagogical note before proceeding to formal statements.

Conceptual orientation

A CenturionDEX v3 pool is a two-asset automated market maker in which every liquidity provider (LP) selects a price interval $[p_a, p_b] \subset (0, \infty)$ over which to deploy capital. A position is characterized by the triple $(p_a,\ p_b,\ L)$, where $L > 0$ is the *liquidity parameter* governing local market depth. While the spot price remains inside the range, the position contributes to trade execution and accrues swap fees in proportion to L; once the price exits the range, the position holds a single asset and ceases to earn fees until the price returns. The square-root price $\sqrt{p}$ is adopted throughout as the natural state coordinate, since token balances, reserves, and swap amounts become affine in $\sqrt{p}$ on any constant-liquidity interval.

Relation to v2-style constant-product AMMs. The v3 design generalizes the classical constant-product automated market maker exemplified by Centurion v2 [1], which enforces a single reserve invariant $x \cdot y = k$ supported on the entire price half-line $(0, \infty)$. In v2, every LP receives identical, fungible exposure to the full curve; the required deposit is always a 50/50 split by value; aggregate depth is uniform across prices; swap fees accumulate as additional reserves, so that $L = \sqrt{k}$ grows mechanically with each trade; and there is exactly one fee tier per pair. CenturionDEX v3 relaxes each of these choices. First, liquidity is *concentrated*: each position's reserves satisfy a translated constant-product relation $(x_v - L/\sqrt{p_b})(y_v - L\sqrt{p_a}) = L^2$ that produces zero withdrawable balances precisely at the interval endpoints, so capital is deployed only where the LP expects price action [2]. Second, the required deposit ratio at entry is determined by the entry price and the range: it is 50/50 only at $p_0 = \sqrt{p_a p_b}$, pure-X below the range, and pure-Y above (§3). Third, positions are non-fungible (NFTs) because two LPs may select different ranges on the same pool, which mandates per-position fee accounting via the outside-growth accumulators $f_o^{\alpha}(i)$ rather than a single pool-wide index.

*https://centurionchain.org

Fourth, fees are paid out of band: they do *not* increase L, a conscious departure from v2 that separates price state from fee accrual and enables the inside-range bookkeeping of §14. Fifth, the pool exposes a discrete family of tiers with tick spacings $s \in \{1, 10, 60, 200\}$ so that LPs can price-discriminate volatility and adverse-selection risk. The practical consequence is a tunable capital-efficiency multiplier $\kappa = L_{\text{v3}}/L_{\text{v2}}$ that can exceed two orders of magnitude for tight ranges, at the cost of stronger path dependence and the operational burden of rebalancing (§19). Conversely, a v3 position with $p_a \to 0$ and $p_b \to \infty$ degenerates exactly into a v2 full-range position, recovering the classical theory as the limiting case.

Notation and terminology

Symbol / term	**Definition**
Assets and prices	
CTN, USDC	The two pool assets; X = CTN (base), Y = USDC (quote).
p	Spot price, expressed as USDC per CTN.
$\sqrt{p}$	Square-root price, the canonical state variable.
$[p_a, p_b]$	Lower and upper price bounds of a position's active range.
Depth and discretization	
L	Liquidity parameter of a single position (local depth).
L_{tot}	Active liquidity at the current price (sum over in-range positions).
Tick index i	Integer price index; $p(i) = 1.0001^i$.
Tick crossing	Event at which L_{tot} changes as the spot moves past an initialized tick.
Position accounting	
LP	Liquidity provider who deposits into a chosen range.
$V(p)$	Mark-to-market value of a position at price p.
$W(p)$	Passive-hold benchmark: $W(p) = x_0 p + y_0$.
$\text{IL}(p)$	Impermanent loss, $V(p)/W(p) - 1$.
Execution and fees	
Price impact	Fractional change in spot induced by a trade.
Slippage	Deviation between quoted and realized execution price.
φ	Pool fee rate (per-swap, charged on the input token).
$f_g^\alpha,\ f_o^\alpha,\ f_r^\alpha$	Global, outside, and in-range fee growth per unit liquidity in token $\alpha \in \{X, Y\}$.
TWAP	Time-weighted average price maintained by the pool oracle.
Protocol fee	Governance-controlled share of φ diverted from LPs to a treasury.
Lifecycle and ancillary operations	
Flash swap	In-transaction borrow-and-repay against pool reserves.
Rebalance	Burn a position and re-mint at a range re-centered on the current spot.

Suggested reading path

A first pass through the document may proceed linearly along the following arc, which develops the mechanics in order of increasing compositional complexity: notation and setup (§1), entry regimes and the deposit map (§3), position value and impermanent loss (§4–5), single-interval swap mechanics (§6), multi-tick execution and the aggregated liquidity curve (§9–11), fee accounting and the TWAP oracle (§13–14), and rebalancing alongside the capital-efficiency comparison (§18–19). Advanced topics — oracle internals, multi-pool routing, position-NFT lifecycle accounting, the protocol-fee switch, and edge-case behavior — are developed in §22–26; the proofs and fixed-point arithmetic appear in the appendices.

1 Notation and setup

Beginner note. This section defines the language used everywhere else. Focus on four symbols: p (price), $[p_a, p_b]$ (your active range), L (liquidity depth), and $\sqrt{p}$ (square-root price used to make formulas linear). If this section is clear, the rest is much easier.

Throughout this document, a CenturionDEX v3 pool holds two tokens $X = \text{CTN}$ and $Y = \text{USDC}$. The spot price is

$$p \;=\; \frac{Y \text{ per pool}}{X \text{ per pool}} \;=\; \text{USDC per CTN.}$$

A concentrated liquidity position is a triple (p_a, p_b, L) with $0 < p_a < p_b$ and liquidity parameter $L > 0$. When the spot price satisfies $p_a \le p \le p_b$, the position contributes virtual reserves

$$x_v(p) \;=\; \frac{L}{\sqrt{p}}, \qquad y_v(p) \;=\; L\sqrt{p},$$

which satisfy the constant-product invariant $x_v\, y_v \;=\; L^2$ and the price identity $y_v/x_v \;=\; p$. The real (withdrawable) balances are obtained by the translation

$$x(p) \;=\; x_v(p) - \frac{L}{\sqrt{p_b}}, \qquad y(p) \;=\; y_v(p) - L\sqrt{p_a},$$

clipped to $[0, \infty)$ outside the interval. Equivalently,

$$x(p) = \begin{cases} L\Big(\frac{1}{\sqrt{p_a}} - \frac{1}{\sqrt{p_b}}\Big), & p \le p_a, \\ L\Big(\frac{1}{\sqrt{p}} - \frac{1}{\sqrt{p_b}}\Big), & p_a \le p \le p_b, \\ 0, & p \ge p_b, \end{cases} \qquad y(p) = \begin{cases} 0, & p \le p_a, \\ L\big(\sqrt{p} - \sqrt{p_a}\big), & p_a \le p \le p_b, \\ L\big(\sqrt{p_b} - \sqrt{p_a}\big), & p \ge p_b. \end{cases} \tag{1}$$

A clean running example. Unless stated otherwise, our worked examples use the position

$$p_a \;=\; 1600, \qquad p_b \;=\; 2500, \qquad L \;=\; 45{,}000,$$

so that $\sqrt{p_a} = 40$, $\sqrt{p_b} = 50$. With entry price $p_0 = 2025$ ($\sqrt{p_0} = 45$), the initial real balances are

$$x_0 \;=\; 45000\big(\tfrac{1}{45} - \tfrac{1}{50}\big) \;=\; 45000 \cdot \tfrac{1}{450} \;=\; 100 \text{ CTN}, \qquad y_0 \;=\; 45000\,(45 - 40) \;=\; 225{,}000 \text{ USDC}.$$

At $p_0 = 2025$ these are worth $V(p_0) = 100 \cdot 2025 + 225{,}000 = 427{,}500$ USDC.

2 Tick math

Beginner note. Prices are continuous in theory, but the protocol stores them on a discrete grid (ticks). This section explains how to convert human prices (like 2025 USDC/CTN) into tick indices that smart contracts actually use.

CenturionDEX v3 discretizes the price continuum by powers of 1.0001:

$$p(i) \;=\; 1.0001^i, \qquad i \in \mathbb{Z}, \qquad \sqrt{p(i)} = 1.0001^{i/2}.$$

The current tick index is

$$i_c \;=\; \big\lfloor \log_{1.0001}(p) \big\rfloor.$$

Example 2.1 (Mapping a price to a tick)**.** *Let $p = 2025$ USDC/CTN. Then*

$$i_c \;=\; \left\lfloor \frac{\ln 2025}{\ln 1.0001} \right\rfloor \;=\; \lfloor 76{,}012.27\ldots \rfloor \;=\; 76{,}012,$$

so $p(76012) \approx 2024.797$ *and* $p(76013) \approx 2025.000$*. A liquidity provider who wants the idealized range* $[1600, 2500]$ *would select the initialized ticks*

$$i_a \;=\; \lceil \log_{1.0001} 1600 \rceil, \qquad i_b \;=\; \lfloor \log_{1.0001} 2500 \rfloor,$$

namely $i_a = 74{,}097$ *(*$p \approx 1600.09$*) and* $i_b = 78{,}244$ *(*$p \approx 2499.91$*). For the arithmetic below we use the idealized prices* 1600 *and* 2500 *exactly.*

Tick spacing. Pools are created with a tick spacing $s \in \{1, 10, 60, 200\}$ that restricts initializable ticks to multiples of s. The coarser the spacing, the fewer initializable ticks and the cheaper the swap path, at the cost of coarser range granularity. In practice:

$$s = 1 \text{ (0.01\% fee tier)}, \quad s = 10 \text{ (0.05\%)}, \quad s = 60 \text{ (0.30\%)}, \quad s = 200 \text{ (1.00\%)}.$$

A tick spacing $s = 60$ corresponds to a minimum price step of $1.0001^{60} \approx 1.006018$, i.e. roughly 0.6%.

Example 2.2 (Rounding to a valid tick)**.** *For the* 0.30% *tier (*$s = 60$*) and target lower price* $p_a = 1600$*, valid tick indices are multiples of* 60*. The unrounded index is* $\log_{1.0001}(1600) \approx 74{,}096.82$*. Rounding up to the next multiple of* 60 *gives* $i_a = 74{,}100$*, which corresponds to* $p \approx 1600.57$*. For* $p_b = 2500$*, rounding down gives* $i_b = 78{,}240$*, or* $p \approx 2499.17$*. The effective range is* [1600.57, 2499.17]*, approximately* 0.04% *narrower than the nominal* $[1600, 2500]$*.*

3 Adding liquidity: three entry regimes

Beginner note. Your deposit composition depends on where the current price is relative to your range. Below range means mostly CTN, above range means mostly USDC, and inside range means a mix of both.

We mint a position with parameters $p_a = 1600$, $p_b = 2500$, $L = 45{,}000$, entered at three different spot prices. In each case the deposit is determined by (1).

Example 3.1 (Regime 1: entry inside the range, $p_0 = 2025$)**.**

$$x_0 \;=\; 45000\left(\tfrac{1}{45} - \tfrac{1}{50}\right) = 100 \text{ CTN}, \qquad y_0 \;=\; 45000\,(45 - 40) = 225{,}000 \text{ USDC}.$$

Value at entry: $V(p_0) = 100 \cdot 2025 + 225{,}000 = 427{,}500$ *USDC.*

Example 3.2 (Regime 2: entry below the range, $p_0 = 1225$)**.** *Since* $p_0 < p_a$*, the deposit is pure CTN:*

$$x_0 \;=\; 45000\left(\tfrac{1}{40} - \tfrac{1}{50}\right) = 45000 \cdot \tfrac{1}{200} = 225 \text{ CTN}, \qquad y_0 = 0.$$

Value at entry: $V = 225 \cdot 1225 = 275{,}625$ *USDC.*

Example 3.3 (Regime 3: entry above the range, $p_0 = 3600$)**.** *Since* $p_0 > p_b$*, the deposit is pure USDC:*

$$x_0 = 0, \qquad y_0 \;=\; 45000\,(50 - 40) = 450{,}000 \text{ USDC}.$$

Value at entry: $V = 450{,}000$ *USDC.*

Dollar-value split at entry inside the range. Inside the range the USDC share of the deposit at $p = p_0$ is

$$\phi_Y(p_0) \;=\; \frac{y_0}{x_0 p_0 + y_0} \;=\; \frac{\sqrt{p_0} - \sqrt{p_a}}{(\sqrt{p_0} - \sqrt{p_a}) + \sqrt{p_0}(1 - \sqrt{p_0}/\sqrt{p_b})}.$$

For the running position at $p_0 = 2025$ this gives $\phi_Y = 225000/427500 \approx 52.63\%$: the provider is *not* required to deposit a 50/50 split. The split is 50/50 only when $p_0 = \sqrt{p_a p_b}$, the geometric mean of the bounds.

Example 3.4 (Geometric-mean deposit)**.** *For* $[p_a, p_b] = [1600, 2500]$*, the 50/50 point is* $p^\star = \sqrt{1600 \cdot 2500} = 2000$*. At* $p^\star$ *with* $L = 45{,}000$*:*

$$x^\star = 45000(1/\sqrt{2000} - 1/50) \approx 45000\,(0.02236 - 0.02000) \approx 106.07 \text{ CTN},$$

$$y^\star = 45000(\sqrt{2000} - 40) \approx 45000\,(44.721 - 40) \approx 212{,}132 \text{ USDC},$$

with $x^\star p^\star \approx 212{,}132 \approx y^\star$*. Equal dollar weights as expected.*

4 Position value $V(p)$

Beginner note. This section answers: "What is my position worth if price moves?" The key idea is that the value curve is piecewise and changes behavior at the range boundaries.

For the running position $(1600, 2500, 45{,}000)$, the piecewise value $V(p) = x(p)p + y(p)$ is

$$V(p) \;=\; \begin{cases} 225\,p, & p \le 1600, \\ 45000\big(2\sqrt{p} - \frac{p}{50} - 40\big), & 1600 \le p \le 2500, \\ 450{,}000, & p \ge 2500. \end{cases}$$

The inside-range formula $V(p) = L\big(2\sqrt{p} - p/\sqrt{p_b} - \sqrt{p_a}\big)$ is obtained by substituting (1) directly.

Example 4.1 (Value table)**.**

$\sqrt{p}$	p	$x(p)$ *(CTN)*	$y(p)$ *(USDC)*	$V(p)$ *(USDC)*	*regime*
40	1600	225.0000	0	360,000	*lower boundary*
42	1764	171.4286	90,000	392,400	*inside*
45	2025	100.0000	225,000	427,500	*entry*
48	2304	37.5000	360,000	446,400	*inside*
50	2500	0.0000	450,000	450,000	*upper boundary*

Worked row at $\sqrt{p} = 48$*:*

$$x(2304) = 45000\big(\tfrac{1}{48} - \tfrac{1}{50}\big) = 45000 \cdot \tfrac{2}{2400} = 37.5, \qquad y(2304) = 45000\,(48 - 40) = 360{,}000,$$

$$V(2304) \;=\; 37.5 \cdot 2304 + 360{,}000 \;=\; 86{,}400 + 360{,}000 \;=\; 446{,}400.$$

Worked row at $\sqrt{p} = 42$*:*

$$x(1764) = 45000\big(\tfrac{1}{42} - \tfrac{1}{50}\big) = 45000 \cdot \tfrac{8}{2100} = \tfrac{1200}{7} \approx 171.4286,$$

$$y(1764) = 45000\,(42 - 40) = 90{,}000, \qquad V = \tfrac{1200}{7} \cdot 1764 + 90{,}000 = 302{,}400 + 90{,}000 = 392{,}400.$$

Concavity. On $[p_a, p_b]$, $V''(p) = -\frac{L}{2p^{3/2}} < 0$, so V is strictly concave. The tangent line to V at p_0 is precisely the passive-hold line $W(p) = x_0 p + y_0$. This gives the clean geometric picture: LP value is a concave curve sitting on the chord connecting the two boundary values, and the passive-hold line is tangent to that curve at p_0.

5 Impermanent loss

Beginner note. Impermanent loss compares two worlds: (1) providing liquidity and (2) doing nothing except holding initial tokens. IL is not always catastrophic; it is a tradeoff against fee income.

For an LP that enters at p_0 with deposit (x_0, y_0), the passive portfolio value is $W(p) = x_0\,p + y_0$, and the impermanent loss fraction is

$$\mathrm{IL}(p) \;=\; \frac{V(p)}{W(p)} - 1.$$

Example 5.1 (IL for the inside-range position, $p_0 = 2025$)**.** *Here* $W(p) = 100\,p + 225{,}000$.

$\sqrt{p}$	p	$V(p)$	$W(p)$	$\mathrm{IL}(p)$
40	1600	360,000	385,000	−6.494%
42	1764	392,400	401,400	−2.242%
45	2025	427,500	427,500	0%
48	2304	446,400	455,400	−1.976%
50	2500	450,000	475,000	−5.263%

Row at $p = 1600$: $V - W = 360{,}000 - 385{,}000 = -25{,}000$; relative $-25000/385000 = -6.4935\%$. Row at $p = 2500$: $V - W = 450{,}000 - 475{,}000 = -25{,}000$; relative -5.2632%.

Example 5.2 (Case 3.1 closed form)**.** *Position minted above the range at $p_0 = 3600$ (Ex. 3.3), price later falls to $p = 1225$ (below the range). The position is now entirely converted to CTN:*

$$x(1225) = 225, \quad y = 0, \quad V = 225 \cdot 1225 = 275{,}625, \quad W = 0 \cdot 1225 + 450{,}000 = 450{,}000.$$

The closed-form prediction is

$$\mathrm{IL}(p) \;=\; \frac{p}{\sqrt{p_a p_b}} - 1 \;=\; \frac{1225}{\sqrt{1600 \cdot 2500}} - 1 \;=\; \frac{1225}{2000} - 1 \;=\; -38.75\%,$$

which matches $V/W - 1 = 275625/450000 - 1 = -0.3875$ exactly.

Example 5.3 (Case 2.3 closed form)**.** *Position minted below the range at $p_0 = 1225$ (Ex. 3.2), price later rises to $p = 3600$ (above the range). The position is entirely converted to USDC:*

$$V = 450{,}000, \qquad W = 225 \cdot 3600 + 0 = 810{,}000.$$

Closed form:

$$\mathrm{IL}(p) \;=\; \frac{\sqrt{p_a p_b}}{p} - 1 \;=\; \frac{2000}{3600} - 1 \;=\; -44.\overline{4}\%,$$

which equals $450000/810000 - 1 = -0.4\overline{4}$.

Range width versus IL. For an inside-range LP entering at p_0, a narrower range means a larger fraction of L goes into each unit of value, so fees per unit value are higher — at the cost of steeper V'' and thus larger IL per unit price move. A useful rule of thumb: halving the range width roughly doubles both fee APR and the worst-case IL at the boundaries.

Example 5.4 (Tight vs wide range)**.** *Compare two positions, both entered at $p_0 = 2025$ with L chosen so that each holds exactly 225,000 USDC of value at entry:*

Range	L	x_0	y_0	*IL at $p = 1600$*
$[1600, 2500]$	45,000	100.00 CTN	225,000 USDC	-6.494%
$[1936, 2116]$	204,545	24.69 CTN	204,545 USDC	*position out of range*

The tight $[1936, 2116]$ position concentrates $\sim 4.5\times$ more liquidity at $p_0 = 2025$, so all else equal it would collect fees at roughly $4.5\times$ the wide position's rate while price stays in-range, but exits to pure CTN immediately upon a drop to $p = 1600$ and stops earning fees.

6 Single-interval swaps: the four cases

Beginner note. All swaps in one active interval follow the same mechanics; only "exact-in" vs "exact-out" and "buy" vs "sell" changes sign conventions. Use this section as your core swap calculator.

Throughout this section the pool hosts only our running position, so effective liquidity is $L_{\mathrm{tot}} = 45{,}000$ as long as the spot stays in $[1600, 2500]$. Swap math (no fees) is governed by the virtual-reserve identity $L_{\mathrm{tot}}(\sqrt{p})$-parametrization, which yields, for a move from $\sqrt{p}_1$ to $\sqrt{p}_2$,

$$\Delta x \;=\; L_{\mathrm{tot}}\Big(\tfrac{1}{\sqrt{p}_2} - \tfrac{1}{\sqrt{p}_1}\Big), \qquad \Delta y \;=\; L_{\mathrm{tot}}\big(\sqrt{p}_2 - \sqrt{p}_1\big).$$

Positive Δy means token Y *enters* the pool; positive Δx means token X *enters* the pool. We start from $\sqrt{p}_1 = 45$ ($p = 2025$) in every case.

Example 6.1 (Case A: exact-out, buy 25 CTN)**.** *User removes* $\Delta x_{\text{out}} = 25$ *CTN. The new* $1/\sqrt{p}_2$ *satisfies*

$$\tfrac{1}{\sqrt{p}_2} = \tfrac{1}{45} + \tfrac{25}{45000} = \tfrac{1000+25}{45000} = \tfrac{1025}{45000},$$

hence $\sqrt{p}_2 = 45000/1025 = \frac{1800}{41} \approx 43.9024$ *and* $p_2 \approx 1927.4242$*. USDC paid in:*

$$\Delta y_{\text{in}} \;=\; 45000\,(\sqrt{p}_1 - \sqrt{p}_2) \;=\; 45000\big(45 - \tfrac{1800}{41}\big) \;=\; 45000 \cdot \tfrac{45}{41} \;=\; \tfrac{2{,}025{,}000}{41} \approx 49{,}390.24.$$

Effective average price per CTN: $49{,}390.24/25 \approx 1975.61$ *USDC/CTN, which lies between the start and end spot prices, as it must.*

Example 6.2 (Case B: exact-in, sell 45,000 USDC)**.** *User adds* $\Delta y_{\text{in}} = 45{,}000$*. Then* $\sqrt{p}_2 = 45 + 45000/45000 = 46$ *and* $p_2 = 2116$*. CTN out:*

$$\Delta x_{\text{out}} \;=\; 45000\big(\tfrac{1}{45} - \tfrac{1}{46}\big) \;=\; 45000 \cdot \tfrac{1}{2070} \;=\; \tfrac{4500}{207} \approx 21.7391 \text{ CTN}.$$

Check: $21.7391 \cdot (\textit{avg price}) \approx 45000$*, with avg price* ≈ 2070*.*

Example 6.3 (Case C: exact-in, sell 50 CTN)**.** *User adds* $\Delta x_{\text{in}} = 50$*. Then*

$$\tfrac{1}{\sqrt{p}_2} = \tfrac{1}{45} + \tfrac{50}{45000} = \tfrac{1050}{45000}, \qquad \sqrt{p}_2 = \tfrac{45000}{1050} = \tfrac{300}{7} \approx 42.8571,$$

$p_2 = (300/7)^2 = \frac{90000}{49} \approx 1836.7347$*. USDC out:*

$$\Delta y_{\text{out}} \;=\; 45000\big(45 - \tfrac{300}{7}\big) \;=\; 45000 \cdot \tfrac{15}{7} \;=\; \tfrac{675{,}000}{7} \approx 96{,}428.57.$$

Example 6.4 (Case D: exact-out, buy 90,000 USDC)**.** *User removes* $\Delta y_{\text{out}} = 90{,}000$*, so* $\sqrt{p}_2 = 45 - 90000/45000 = 43$*,* $p_2 = 1849$*. CTN in:*

$$\Delta x_{\text{in}} \;=\; 45000\big(\tfrac{1}{43} - \tfrac{1}{45}\big) \;=\; 45000 \cdot \tfrac{2}{1935} \;=\; \tfrac{90000}{1935} \;=\; \tfrac{2000}{43} \approx 46.5116 \text{ CTN}.$$

7 Price impact and slippage

Beginner note. Price impact and slippage are related but not identical. Price impact is market movement caused by your size; slippage is what you personally experience versus your expectation/tolerance settings.

For an exact-in swap of size Δy_{in} starting at price p_1, the end price and average execution price are

$$\sqrt{p}_2 = \sqrt{p}_1 + \frac{\Delta y_{\text{in}}}{L_{\text{tot}}}, \qquad \bar{p} \;\equiv\; \frac{\Delta y_{\text{in}}}{\Delta x_{\text{out}}} \;=\; \sqrt{p}_1\,\sqrt{p}_2.$$

The price-impact fraction (end spot vs start spot) is

$$\eta \;=\; \frac{p_2}{p_1} - 1 \;=\; \left(1 + \frac{\Delta y_{\text{in}}}{L_{\text{tot}}\sqrt{p}_1}\right)^2 - 1.$$

Example 7.1 (Price impact of a 45,000 USDC buy)**.** *Using Case B (Ex. 6.2):* $\sqrt{p}_1 = 45$*,* $\sqrt{p}_2 = 46$*,* $\bar{p} = 45 \cdot 46 = 2070$*. Impact* $\eta = (46/45)^2 - 1 \approx 4.494\%$*. The average execution price is* 2070 *vs start spot* 2025*, a* 2.22% *premium over spot — about half the end-to-end impact, the canonical result for a constant-liquidity move.*

Example 7.2 (Slippage tolerance of 1%)**.** *A trader submits a Case B buy with* 1% *slippage tolerance, meaning the minimum CTN out must satisfy*

$$\Delta x_{\text{out,min}} \;=\; 0.99 \cdot \frac{\Delta y_{\text{in}}}{p_1} \;=\; 0.99 \cdot \frac{45{,}000}{2025} \;\approx\; 22.0000 \text{ CTN}.$$

The actual output from Ex. 6.2 is 21.7391 *CTN, which is* below *the minimum. The transaction would revert. To clear, the trader would either reduce size (e.g.* $\Delta y_{\text{in}} = 20{,}000$ *gives* $\sqrt{p}_2 \approx 45.4444$*, output* ≈ 9.901 *CTN, slippage vs spot* $\approx 0.22\%$*, passes) or raise tolerance.*

8 A swap with a 0.30% fee

Beginner note. Fees do not move price directly; only the net amount after fee moves price. This section separates "fee paid" from "effective traded amount" so accounting stays correct.

Let the pool fee be $\varphi = 0.003$. For an exact-in swap, the effective amount that moves the price is $(1-\varphi)\Delta_{\text{in}}$ and the fee retained by LPs is $\varphi\Delta_{\text{in}}$.

Example 8.1 (Sell 100 CTN with 0.30% fee)**.** *Gross in:* $\Delta x_{\text{in}} = 100$. *Fee:* 0.30 *CTN retained for LPs. Effective in:* 99.70 *CTN.*

$$\tfrac{1}{\sqrt{p}_2} = \tfrac{1}{45} + \tfrac{99.70}{45000} = \tfrac{1000+99.70}{45000} = \tfrac{1099.70}{45000},$$

hence $\sqrt{p}_2 = 45000/1099.70 \approx 40.9203$ *and* $p_2 \approx 1674.47$. *USDC out:*

$$\Delta y_{\text{out}} \;=\; 45000\big(45 - 40.9203\big) \;\approx\; 45000 \cdot 4.0797 \;\approx\; 183{,}587.$$

Average execution price per CTN: $183{,}587/100 \approx 1835.87$ *USDC/CTN, below* $p_1 = 2025$ *as expected for a seller moving the price down.*

Example 8.2 (Fee growth per unit liquidity)**.** *The* 0.30 *CTN fee from Ex. 8.1 is charged in the input token (X), so it contributes to* f_g^X. *With* $L_{\text{tot}} = 45{,}000$ *during the swap:*

$$\Delta f_g^X \;=\; \frac{0.30}{45{,}000} \;=\; 6.6\overline{6} \times 10^{-6} \;\; \textit{CTN per unit } L.$$

A 1 *L position that stays fully in-range accrues* $6.6\overline{6}\mu$*CTN from this swap.*

9 Multi-tick swap: crossing an initialized boundary

Beginner note. When price crosses a boundary, active liquidity changes abruptly. So one user trade can break into multiple legs, each with different depth and therefore different marginal price movement.

Now the pool has *two* stacked positions:

$$\text{Pos 1: } (p_a, p_b, L) = (1600, 2500, 45{,}000), \qquad \text{Pos 2: } (2500, 3600, 30{,}000).$$

Inside each range the active liquidity is the corresponding L; at the shared boundary $p = 2500$ the liquidity jumps from 45,000 to 30,000 as the spot moves upward. Assume the spot price starts at $p_1 = 2025$ ($\sqrt{p}_1 = 45$) and a trader submits an exact-out buy of 200 CTN.

Example 9.1 (Multi-tick buy of 200 CTN)**.** ***Leg 1.*** *Drain Pos 1 up to* $\sqrt{p} = 50$ *(*$p = 2500$*):*

$$\Delta x_{\text{out}}^{(1)} = 45000\big(\tfrac{1}{45} - \tfrac{1}{50}\big) = 100 \text{ CTN}, \qquad \Delta y_{\text{in}}^{(1)} = 45000\,(50 - 45) = 225{,}000 \text{ USDC}.$$

So 100 *CTN are delivered and* 225,000 *USDC are paid to cross tick* $p = 2500$.

Leg 2. *The remaining* 100 *CTN are filled inside Pos 2 (*$L = 30{,}000$*), starting at* $\sqrt{p} = 50$*:*

$$\tfrac{1}{\sqrt{p}_2} = \tfrac{1}{50} - \tfrac{100}{30000} = \tfrac{600-100}{30000} = \tfrac{500}{30000} = \tfrac{1}{60},$$

so $\sqrt{p}_2 = 60$, $p_2 = 3600$. *USDC paid in:*

$$\Delta y_{\text{in}}^{(2)} = 30000\,(60 - 50) = 300{,}000 \text{ USDC}.$$

Totals. 200 *CTN bought for* $225{,}000+300{,}000 = 525{,}000$ *USDC. Average execution price* 2625 *USDC/CTN, which correctly exceeds the starting spot* 2025 *and lies below the final spot* 3600.

10 A deeper multi-tick swap: two upward crossings

Beginner note. This section shows a practical limit: a large order may run out of liquidity before consuming full input size. That is why routers split routes and set strict execution constraints.

Add a third stacked position on top of Pos 2:

$$\text{Pos 3: } (p_a, p_b, L) = (3600, 4900, 14{,}000),$$

so that crossing $p = 3600$ upward drops active liquidity from 30,000 to 14,000. A trader submits an exact-in sell of $\Delta y_{\text{in}} = 900{,}000$ USDC starting from $\sqrt{p} = 45$.

Example 10.1 (Three-leg upward swap)**.** ***Leg 1.*** *From* $\sqrt{p} = 45$ *to* $\sqrt{p} = 50$ *at* $L_{\text{tot}} = 45{,}000$ *consumes* $\Delta y^{(1)} = 45000(50 - 45) = 225{,}000$ *USDC and delivers* $45000(1/45 - 1/50) = 100$ *CTN. Remaining USDC budget* 675,000.

Leg 2. *From* $\sqrt{p} = 50$ *to* $\sqrt{p} = 60$ *at* $L_{\text{tot}} = 30{,}000$ *consumes* $\Delta y^{(2)} = 30000(60 - 50) = 300{,}000$ *USDC and delivers* $30000(1/50 - 1/60) = 30000 \cdot (1/300) = 100$ *CTN. Remaining USDC budget* 375,000.

Leg 3. *From* $\sqrt{p} = 60$ *at* $L_{\text{tot}} = 14{,}000$ *with* 375,000 *USDC in. End square-root price satisfies*

$$\sqrt{p}_3 = 60 + \frac{375{,}000}{14{,}000} = 60 + \frac{375}{14} = \frac{840 + 375}{14} = \frac{1215}{14} \approx 86.79, \qquad p_3 \approx 7{,}532.$$

Wait — p_3 exceeds Pos 3's upper bound 4900*. The trader would run out of liquidity before spending the full* 375,000*. Check the last boundary:*

$$\Delta y^{(3,\text{cap})} = 14000(70 - 60) = 140{,}000 \text{ USDC},$$
$$\Delta x^{(3,\text{cap})} = 14000(1/60 - 1/70) = 14000 \cdot \tfrac{1}{420} = \tfrac{100}{3} \approx 33.33 \text{ CTN}.$$

So Leg 3 only absorbs 140,000 *USDC for* 33.33 *CTN before hitting* $\sqrt{p} = 70$ *($p = 4900$). Beyond that, no more liquidity remains in the pool; the remaining* 235,000 *USDC of the order cannot execute and the transaction reverts (or, for exact-in routers, partially fills up to* 465,000 *USDC with* $\sqrt{p} \to 70^-$ *and returns). This illustrates the* liquidity-cap *regime.*

11 Multi-position aggregation and the active liquidity curve

Beginner note. In real pools, many LPs overlap. Effective liquidity at any price is the sum of all active positions at that price. Think of this as a step function of depth across price.

Now consider three overlapping positions in the same pool:

$$\text{A} = (1600, 2500,\ 18{,}000), \quad \text{B} = (2025, 3025,\ 22{,}000), \quad \text{C} = (2500, 3600,\ 30{,}000).$$

By the aggregation proposition, at any price p not on an initialized boundary the effective liquidity is

$$L_{\text{tot}}(p) = \sum_{j:\, p \in [p_{a,j}, p_{b,j}]} L_j.$$

Example 11.1 (Piecewise active liquidity)**.**

$$L_{\text{tot}}(p) = \begin{cases} 0, & p < 1600, \\ 18{,}000, & 1600 \le p < 2025, \\ 40{,}000, & 2025 \le p < 2500, \\ 52{,}000, & 2500 \le p < 3025, \\ 30{,}000, & 3025 \le p \le 3600, \\ 0, & p > 3600. \end{cases}$$

At $p = 2400$ only A and B are active, so $L_{\text{tot}} = 18000 + 22000 = 40000$. At $p = 2800$ all three conditions fail for A (since $p > 2500$), but B and C are both active, giving $22000 + 30000 = 52000$.

Net liquidity values at the five boundary ticks. Let $t_1 = 1600, t_2 = 2025, t_3 = 2500, t_4 = 3025, t_5 = 3600$. Moving from low to high prices:

$$\Delta L(t_1) = +18000, \quad \Delta L(t_2) = +22000, \quad \Delta L(t_3) = +30000 - 18000 = +12000,$$

$$\Delta L(t_4) = -22000, \quad \Delta L(t_5) = -30000.$$

Running sum: 18000, 40000, 52000, 30000, 0, matching Ex. 11.1.

Gross liquidity L_g. Summing liquidity of all positions that touch each tick (regardless of side):

$$L_g(t_1) = 18000,\ L_g(t_2) = 22000,\ L_g(t_3) = 18000 + 30000 = 48000,\ L_g(t_4) = 22000,\ L_g(t_5) = 30000.$$

Gross liquidity governs the maximum-liquidity cap enforced per tick; net liquidity governs swap math.

Example 11.2 (Swap across three boundaries in the aggregated pool)**.** *From $p_1 = 2025$ ($\sqrt{p}_1 = 45$), a trader sells Δy_{in} USDC to push the price all the way to $p_2 = 3025$ ($\sqrt{p}_2 = 55$).*

Leg $[45, 50]$, $L_{\text{tot}} = 40{,}000$: $\Delta y = 40000 \cdot 5 = 200{,}000$ USDC, $\Delta x = 40000(1/45 - 1/50) = 40000 \cdot \frac{1}{450} \approx 88.889$ CTN.

Leg $[50, 55]$, $L_{\text{tot}} = 52{,}000$: $\Delta y = 52000 \cdot 5 = 260{,}000$ USDC, $\Delta x = 52000(1/50 - 1/55) = 52000 \cdot \frac{5}{2750} \approx 94.545$ CTN.

Totals. *$\Delta y_{\text{in}} = 460{,}000$ USDC, $\Delta x_{\text{out}} \approx 183.434$ CTN. Average price ≈ 2507.7 USDC/CTN. Note that the upward boundary at $p = 2500$ increased L_{tot} (from 40,000 to 52,000), so the second leg moves the price more cheaply per USDC than a naive extrapolation of the first leg would suggest.*

12 Range (limit) orders

Beginner note. A one-sided concentrated position can emulate a limit order. Unlike a classical order book fill, execution is distributed across a range and usually earns fees during conversion.

A one-sided concentrated liquidity position acts as a limit order: the LP deposits pure Y (resp. pure X) above (resp. below) the current price, and the position converts to the other token as the price sweeps through the range.

Example 12.1 (Sell-CTN limit order)**.** *Current spot $p_0 = 2025$. An LP wants to sell 100 CTN at an average price of at least 2300. She mints a position with*

$$p_a = 2200, \quad p_b = 2400, \quad y_0 = 0,$$

depositing only CTN. The liquidity is found from the x-only boundary formula

$$x_0 \ = \ L\Big(\tfrac{1}{\sqrt{p_a}} - \tfrac{1}{\sqrt{p_b}}\Big), \qquad \sqrt{p_a} = \sqrt{2200} \approx 46.904, \ \ \sqrt{p_b} = \sqrt{2400} \approx 48.990,$$

$$\tfrac{1}{\sqrt{p_a}} - \tfrac{1}{\sqrt{p_b}} \ \approx \ 0.021320 - 0.020412 \ = \ 9.081 \times 10^{-4},$$

so $L \approx 100/9.081 \times 10^{-4} \approx 110{,}121$. If the price later rises past $p_b = 2400$, the full 100 CTN has been sold for

$$y_0^{final} \ = \ L\left(\sqrt{p_b} - \sqrt{p_a}\right) \ \approx \ 110{,}121 \cdot 2.086 \ \approx \ 229{,}740 \text{ USDC},$$

average fill price ≈ 2297.4 USDC/CTN, between p_a and p_b (it equals $\sqrt{p_a p_b} = \sqrt{2200 \cdot 2400} \approx 2297.83$, the geometric mean). Plus any fees earned while crossing the range.

Comparison with a book-style limit order. A book limit at 2300 fills the entire 100 CTN at exactly 2300 once touched. The range-order variant spreads execution across $[2200, 2400]$ at average ≈ 2297.83, fills *and* earns fees while the price traverses the range, but is also partially filled if price only enters mid-range and reverses. The LP must burn the position once filled to avoid having the conversion reverse on a price pullback.

13 TWAP oracle

Beginner note. TWAP smooths short-term noise by averaging price over time, not just taking the latest trade. This helps protect downstream protocols from brief manipulation spikes.

CenturionDEX v3 maintains a per-pool ring buffer of observations $(t_k,\, a_k)$ where $a_k = \sum_{j<k}(t_j - t_{j-1}) \cdot i_{c,j}$ is the cumulative tick-index-time integral. The arithmetic-mean tick over $[t_0, t_1]$ is

$$\bar{i} \;=\; \frac{a_1 - a_0}{t_1 - t_0}, \qquad \bar{p} \;=\; 1.0001^{\bar{i}}.$$

Example 13.1 (TWAP over a three-segment window)**.** *Over* 600 *seconds the spot tick took the following values:*

Segment	*Duration (s)*	*Tick* i_c
$[0, 200]$	200	76,012
$[200, 450]$	250	76,500
$[450, 600]$	150	77,200

Time-weighted average tick:

$$\begin{aligned}\bar{i} &= \frac{200 \cdot 76012 + 250 \cdot 76500 + 150 \cdot 77200}{600} \\ &= \frac{15{,}202{,}400 + 19{,}125{,}000 + 11{,}580{,}000}{600} \\ &= \frac{45{,}907{,}400}{600} \;=\; 76{,}512.33,\end{aligned}$$

giving $\bar{p} = 1.0001^{76512.33} \approx 2127.63$ *USDC/CTN. Note the arithmetic mean of tick indices corresponds to the* geometric *mean of prices, which is the natural average for log-normal-like price processes.*

Manipulation resistance. To push the TWAP by 1% over a 600-second window, an attacker must hold the spot price at $1.0001^{100} \approx 1.01005$ times the true price for the full window — every block another trader can arbitrage against them, at the pool's full depth. For the depth in Ex. 10.1 (full-stack $\sim$ \$1M), the cost of sustaining a 1% TWAP dislocation for 10 minutes scales with the integral of arbitrage-closure flows and typically exceeds the size of any practical attack.

14 Fee accounting: global, outside, and inside

Beginner note. This is the hardest accounting section. The intuition: the pool tracks fee growth globally, then uses "outside" snapshots at range boundaries to isolate exactly what happened "inside" your range.

Let f_g^X, f_g^Y denote cumulative fees *per unit of liquidity* since pool genesis. At each initialized tick i the pool stores $f_o^\alpha(i)$, the cumulative fee growth on the *outside* of tick i, updated on each crossing by

$$f_o^\alpha(i) \;\leftarrow\; f_g^\alpha - f_o^\alpha(i), \qquad \alpha \in \{X, Y\}.$$

The fee growth inside a range $[i_l, i_u]$ is computed from the global and two outsides according to the current tick i_c:

$$f^\alpha_{\text{above}}(i_u) = \begin{cases} f_o^\alpha(i_u), & i_c < i_u, \\ f_g^\alpha - f_o^\alpha(i_u), & i_c \geq i_u, \end{cases} \qquad f^\alpha_{\text{below}}(i_l) = \begin{cases} f_g^\alpha - f_o^\alpha(i_l), & i_c < i_l, \\ f_o^\alpha(i_l), & i_c \geq i_l, \end{cases}$$

$$f_r^\alpha(i_l, i_u) \;=\; f_g^\alpha - f^\alpha_{\text{below}}(i_l) - f^\alpha_{\text{above}}(i_u).$$

Example 14.1 (Three fee events, one tick crossing)**.** *Consider a pool with only two boundary ticks $t_a = 1600$ and $t_b = 2500$, the running position $(1600, 2500, 45{,}000)$ (so $L_{\text{tot}} = 45000$ while $1600 < p < 2500$), and an extra position C with $p_a = 2500$ that makes the crossing at t_b meaningful. Start with $f_g^Y = 0$ and i_c below t_b but above t_a. Assume the pool charges fees in the input token, and track f_g^Y.*

Event 1. *A trader sells USDC inside the range; 90 USDC of fee accrues with $L_{\text{tot}} = 45000$:*

$$\Delta f_g^Y = \tfrac{90}{45000} = 0.002, \qquad f_g^Y : \ 0 \to 0.002.$$

Event 2. *Price crosses t_b upward, entering a region of $L_{\text{tot}} = 30000$. Outside variables at t_b flip:*

$$f_o^Y(t_b) : \ 0 \to f_g^Y - 0 = 0.002.$$

A subsequent swap in the upper region generates 60 USDC of fee at $L_{\text{tot}} = 30000$:

$$\Delta f_g^Y = \tfrac{60}{30000} = 0.002, \qquad f_g^Y : \ 0.002 \to 0.004.$$

Event 3. *Price crosses t_b downward. Outside variables flip again:*

$$f_o^Y(t_b) : \ 0.002 \to f_g^Y - 0.002 = 0.004 - 0.002 = 0.002.$$

(Same numerical value here because of symmetric accumulation.) A further swap in the lower region with $L_{\text{tot}} = 45000$ generates 90 USDC:

$$\Delta f_g^Y = \tfrac{90}{45000} = 0.002, \qquad f_g^Y : \ 0.004 \to 0.006.$$

Example 14.2 (Collecting fees for the running position)**.** *With i_c currently below t_b and above t_a, and $f_g^Y = 0.006$, $f_o^Y(t_a) = 0$, $f_o^Y(t_b) = 0.002$:*

$$f_{\text{below}}^Y(t_a) = f_o^Y(t_a) = 0, \qquad f_{\text{above}}^Y(t_b) = f_o^Y(t_b) = 0.002,$$

$$f_r^Y(t_a, t_b) = 0.006 - 0 - 0.002 = 0.004.$$

The running position's fee entitlement in USDC since mint is therefore

$$L \cdot \left(f_r^Y - f_r^Y(\textit{at mint})\right) = 45000 \cdot (0.004 - 0) = 180 \text{ USDC}.$$

This equals the $90 + 90 = 180$ USDC of fees that accrued while the spot price was inside the position's range, as it must.

Example 14.3 (Two overlapping collectors)**.** *Continuing Ex. 14.1, suppose a second LP minted position D= $(2025, 2500, 22{,}500)$ at the moment when f_g^Y first reached 0.002 (after Event 1 and before the first tick crossing). The mint-time snapshot for D is $f_r^Y(D, \textit{mint}) = 0.002 - 0 - 0 = 0.002$.*

After Event 3, $f_r^Y(D, \textit{now}) = 0.006 - 0 - 0.002 = 0.004$. D's entitlement is

$$L_D \cdot (0.004 - 0.002) = 22500 \cdot 0.002 = 45 \text{ USDC}.$$

D sees only the fees accrued in its range after its mint*: the 90 USDC of Event 3 is distributed across $L_{\text{tot}} = 45{,}000$, and D's share is $22500/45000 = 1/2$, giving 45 USDC. Checks.*

15 Protocol fees

Beginner note. Not all swap fees go to LPs. This section shows the split between LP revenue and protocol treasury revenue and how that changes LP per-liquidity earnings.

Each pool carries a protocol fee $\psi \in \{0, 1/4, 1/5, 1/6, \ldots, 1/10\}$ applied to the LP fee. If $\psi = 1/6$ on the 0.30% tier, then of every 1 unit of input, 0.003 is fee and $0.003 \cdot (1/6) = 0.0005$ goes to protocol reserves, with the remaining 0.0025 to LPs.

Example 15.1 (Protocol fee on the 100 CTN sell)**.** *From Ex. 8.1, the total fee is 0.30 CTN. With $\psi = 1/6$:*

$$\textit{protocol} \ = \ 0.30 \cdot \tfrac{1}{6} = 0.05 \text{ CTN}, \qquad \textit{LPs} \ = \ 0.30 - 0.05 = 0.25 \text{ CTN}.$$

The LP fee per unit liquidity is now $\Delta f_g^X = 0.25/45000 \approx 5.556 \times 10^{-6}$, down from 6.667×10^{-6} without the protocol fee. Protocol reserves are withdrawable only by the factory owner.

16 Flash swaps

Beginner note. Flash swaps are atomic short-term loans from pool reserves. They are useful for arbitrage and refinancing, but repayment plus fee must happen in the same transaction or everything reverts.

A flash swap lends pool reserves to the caller within a single transaction; the caller must return the owed amount plus fee by the end of the callback. Let the requested amount be Δx_{loan} (CTN). The caller returns $\Delta x_{\text{return}} \geq \Delta x_{\text{loan}}(1+\varphi)$ or its USDC-denominated equivalent at the post-swap spot price, otherwise the transaction reverts.

Example 16.1 (Flash borrow 50 CTN, arbitrage on external venue)**.** *Pool fee* $\varphi = 0.003$*. The caller borrows* $\Delta x_{\text{loan}} = 50$ *CTN, sells them on an external venue at* 2050 *USDC/CTN for* 102,500 *USDC, then must return* $50 \cdot (1.003) = 50.15$ *CTN worth of value to the CenturionDEX pool. The minimum USDC needed to return in-kind at the post-callback pool spot price* p_2 *(assume still* ≈ 2025 *since no pool swap occurred) is*

$$50.15 \cdot 2025 \;\approx\; 101{,}554 \text{ USDC}.$$

Callback surplus: $102{,}500 - 101{,}554 = 946$ *USDC. Net arb profit* ≈ 946 *USDC, less gas. If the external venue moves during execution and the caller cannot meet the return amount, the entire transaction reverts atomically.*

17 Burning a position (partial and full)

Beginner note. Burning is withdrawal. Partial burn scales balances linearly with removed liquidity; full burn exits completely and realizes your current inventory composition (which can be mostly one token).

Example 17.1 (Burn 50% of the running position at $p = 2304$)**.** *State:* $L = 45000$*,* $p_a = 1600$*,* $p_b = 2500$*, current* $\sqrt{p} = 48$*. Real balances:*

$$x = 45000\left(\tfrac{1}{48} - \tfrac{1}{50}\right) = 37.5 \text{ CTN}, \qquad y = 45000\,(48-40) = 360{,}000 \text{ USDC}.$$

Burning 50% *reduces* L *to* 22,500*. The tokens returned to the LP are proportional:*

$$x_{ret} = 18.75 \text{ CTN}, \qquad y_{ret} = 180{,}000 \text{ USDC}.$$

Value of the withdrawal at $p = 2304$*:* $18.75 \cdot 2304 + 180{,}000 = 43{,}200 + 180{,}000 = 223{,}200$ *USDC.*

Example 17.2 (Full burn at $p = 1600$)**.** *With* $\sqrt{p} = 40$ *the position is fully converted to CTN:*

$$x = 45000\left(\tfrac{1}{40} - \tfrac{1}{50}\right) = 225 \text{ CTN}, \qquad y = 0.$$

The LP receives 225 *CTN, worth* $225 \cdot 1600 = 360{,}000$ *USDC. Compared with entry at* $p_0 = 2025$ *where the LP deposited* 100 *CTN +* 225,000 *USDC (=* 427,500 *USDC), the realized loss is* 67,500 *USDC in USDC units, i.e.* -15.79% *relative to deposit. The passive hold of* 100 *CTN +* 225,000 *USDC at* $p = 1600$ *would be worth* $100 \cdot 1600 + 225000 = 385{,}000$ *USDC, so impermanent loss versus hold is*

$$\text{IL} \;=\; \frac{360{,}000}{385{,}000} - 1 \;=\; -6.494\%,$$

matching the first row of the IL table in Ex. 5.1.

18 Rebalancing strategies

Beginner note. Rebalancing is not free: you pay spread/slippage and potentially taxes/fees. The benefit is restoring in-range fee earning. This section frames that cost-vs-benefit decision.

When the price drifts to the edge of a range, the LP faces the choice of (i) leaving the position idle, (ii) burning and re-minting a new range centered on the new spot, or (iii) actively hedging with an external instrument. We work a small numerical comparison.

Example 18.1 (Burn-and-remint at $p = 2500$)**.** *The running position reaches $\sqrt{p} = 50$ after an upward swap (end of Leg 1 in Ex. 9.1). At the boundary the LP holds* 0 *CTN +* 450,000 *USDC. She burns fully and re-mints a new range* $[2000, 3125]$ *centered on the new spot* $p_0' = 2500$*:*

$$p_a' = 2000,\ p_b' = 3125,\ \sqrt{p_a}' \approx 44.721,\ \sqrt{p_b}' \approx 55.902,\ \sqrt{p_0}' = 50.$$

Required split at $p_0' = 2500$ *with, say, L' chosen to redeploy exactly* 450,000 *USDC of value:*

$$\frac{y_0'}{V} = \frac{\sqrt{p_0}' - \sqrt{p_a}'}{\sqrt{p_0}' - \sqrt{p_a}' + \sqrt{p_0}'(1 - \sqrt{p_0}'/\sqrt{p_b}')} = \frac{50 - 44.721}{(50 - 44.721) + 50(1 - 50/55.902)} = \frac{5.279}{5.279 + 5.279} = 0.5,$$

so $x_0' p_0' = y_0' = 225{,}000$*, giving* $x_0' = 90$ *CTN,* $y_0' = 225{,}000$ *USDC. The LP must convert* 225,000 *USDC into* 90 *CTN at the prevailing spot (paying further fees/slippage) before minting. Net effect: the LP has re-centered her range at the new spot and is once again collecting fees on two-sided depth.*

Rebalancing cost vs IL savings. Rebalancing converts unrealized IL into realized P&L plus slippage cost on the rebalance trade. For small, frequent price moves within a band, rebalancing is usually net-negative; for larger moves that threaten to exit the range entirely, it is often positive. A practical heuristic: rebalance only if the price has moved by more than 2· range half-width, and never within the same block (MEV exposure).

19 Capital efficiency vs full-range

Beginner note. Concentrated liquidity can be dramatically more efficient in-range than full-range liquidity, but this advantage disappears when price spends long periods outside your band.

A full-range constant-product (v2-style) pool with the same dollar deposit at entry $p_0 = 2025$ requires $x_0 p_0 = y_0$, i.e. a 50/50 split by value. With $V_0 = 427{,}500$ USDC we have $y_0 = 213{,}750$ USDC, $x_0 = 105.556$ CTN, $k = x_0 y_0$, and $L_{\text{v2}} = \sqrt{k} = \sqrt{105.556 \cdot 213{,}750} \approx \sqrt{22{,}562{,}500} \approx 4{,}750$. This is deployed over $[0, \infty)$ in price.

The running v3 position has $L = 45{,}000$ on $[1600, 2500]$ — roughly 9.47× the v2 liquidity, concentrated in ∼44% of the log-price range around the entry. The concentration factor is

$$\kappa \;=\; \frac{L_{\text{v3}}}{L_{\text{v2}}} \;\approx\; 9.47\times,$$

meaning that, in-range, the v3 position earns fees at 9.47× the rate a v2 position earning the same percentage fee on the same dollar deposit would earn, and offers price impact $1/9.47 \approx 10.56\%$ of the v2 equivalent.

Example 19.1 (Slippage comparison on a $45,000 buy)**.** *Under v3 with* $L_{\text{tot}} = 45{,}000$*, the* 45,000 *USDC buy (Ex. 6.2) moves the spot from* 2025 *to* 2116*, a* 4.494% *price impact.*

Under v2 with $L_{\text{v2}} \approx 4{,}750$ *and deposit value* 427,500 *USDC,* $\sqrt{p}_2 = 45 + 45000/4750 \approx 54.474$*, so* $p_2 \approx 2967.4$ *— a price impact of* ∼ 46.5%*. The v3 pool delivers roughly an order of magnitude better execution on this size, consistent with* $\kappa \approx 9.47$*.*

Caveat: inventory vs throughput. Capital efficiency is a two-edged sword. The v3 position offers ∼ 10× throughput *but only while the price stays in-range.* Outside the range the position earns no fees; a v2 position earns at all prices (at a much lower rate). The in-range fee APR of v3 must be discounted by the fraction of time the price spends in-range to make a fair comparison to v2.

20 End-to-end example: life of a position

Beginner note. If sections feel disconnected, start here. This walkthrough links minting, swaps, fee accrual, crossing, collection, and exit into one continuous LP lifecycle story.

We tie the pieces together with a short narrative.

(a) **Mint.** Alice opens the position $(1600, 2500, L = 45000)$ at spot $p_0 = 2025$ depositing 100 CTN + 225,000 USDC. Pool state: i_c inside range, $f_g^Y = 0$, $f_o^Y(t_a) = 0$, $f_o^Y(t_b) = 0$ (initialized with $i \le i_c \Rightarrow f_o^Y(i) = f_g^Y$; here $t_b > i_c$ so $f_o^Y(t_b) = 0$).

(b) **Swap.** A trader sells 45,000 USDC into the pool (Case B, Sec. 6); the spot rises to $p = 2116$. With $\varphi = 0.003$ on the input, the fee is 135 USDC, so f_g^Y grows by $135/45000 = 0.003$.

(c) **Crossing.** A larger buy of 200 CTN (Sec. 9) lifts the spot across $t_b = 2500$ into Pos 2's range. At the crossing, $f_o^Y(t_b) \leftarrow f_g^Y - f_o^Y(t_b)$.

(d) **TWAP snapshot.** An external contract reads the 10-minute TWAP and observes a tick consistent with ≈ 2400 USDC/CTN, reflecting the weighted mean of the pre-crossing and post-crossing regimes.

(e) **Collect.** Alice calls `collect`. Her share of fees is $L \cdot f_r^Y$ evaluated between her mint snapshot and now.

(f) **Rebalance or burn.** With the spot now above her range ($p > 2500$), Alice either (i) burns and re-mints around the new level, or (ii) waits for a mean-revert. She burns at $p = 2304$ after the price retraces, receiving 37.5 CTN + 360,000 USDC plus her accrued fees. Her total USDC-denominated value is $V(2304) + \text{fees} = 446{,}400 + \text{fees}$, versus a passive hold worth $W(2304) = 100 \cdot 2304 + 225000 = 455{,}400$, so the LP's outcome relative to hold is $+\text{fees} - 9{,}000$ USDC. She profits versus hold iff fees exceed 9,000 USDC.

21 Theory layer from the research literature

Beginner note. This section adds the "why" behind the formulas. It links the practical equations in this document to the formal AMM literature so readers can distinguish implementation facts from broader theoretical results.

A. CFMMs as feasible-set geometry

Modern AMM analysis treats a pool as a state vector of reserves and a rule that accepts or rejects net reserve changes. A convenient abstraction is a nondecreasing, concave trading function φ that defines the feasible trade set for current reserves $r \in \mathbb{R}_+^n$:

$$\mathcal{T}(r) \;=\; \{\, q \in \mathbb{R}^n \mid r + q \ge 0,\ \varphi(r+q) \ge \varphi(r) \,\}.$$

For two assets and the constant-product family, this recovers the familiar curve (with fees handled as an input adjustment) and marginal price from reserve ratios [1, 3]. The broader convex-analysis view explains why many routing and execution tasks can be solved as convex programs in multi-asset settings [5], and why CFMMs admit canonical geometric representations beyond differentiable special cases [7].

B. Concentrated liquidity as piecewise depth

Centurion v3 can be viewed as a superposition of many finite-interval constant-product fragments [2]. If $s = \sqrt{p}$ and positions are indexed by j with depths L_j and ranges $[p_{a,j}, p_{b,j}]$, then effective depth in square-root space is

$$L_{\text{eff}}(s) \;=\; \sum_j L_j \, \mathbf{1}\{ s \in [\sqrt{p_{a,j}}, \sqrt{p_{b,j}}] \}\,.$$

This yields the integral form of cross-tick execution used implicitly throughout this note:

$$\Delta y \;=\; \int_{s_1}^{s_2} L_{\text{eff}}(s)\, ds, \qquad \Delta x \;=\; \int_{s_1}^{s_2} L_{\text{eff}}(s)\, d\!\left(\frac{1}{s}\right).$$

Inside one interval, these collapse to the closed forms already used earlier. Across many initialized ticks, they produce the piecewise-leg swap decomposition and explain why crossing behavior is deterministic once the liquidity ladder is known [2].

C. Arbitrage, price tracking, and oracle logic

CFMM spot prices are not externally forced; they are corrected by arbitrage. Under standard assumptions, arbitrage keeps AMM prices close to reference markets [3]. This is the economic backbone of TWAP oracles: cumulative price statistics become informative because deviations are monetizable and therefore short-lived in competitive markets [4, 1, 2]. In implementation terms, this is why v2/v3 rely on cumulative observations over windows rather than instantaneous spot reads.

D. LP payoff structure and adverse selection

From the LP perspective, concentrated-liquidity payoffs are state-contingent and can be studied as engineered payoff surfaces [6]. A practical decomposition over horizon H is

$$\text{LP Edge}_H \;\approx\; \text{Fees}_H - \text{IL}_H - \text{LVR}_H - \text{Gas/Rebalance}_H,$$

where LVR (loss-versus-rebalancing) captures stale-price adverse selection against LPs [10]. This decomposition clarifies a common confusion: IL is not the only drag. Even with low headline IL, adverse selection and execution frictions can dominate if volatility and informed flow are high.

E. Execution microstructure and MEV-aware costs

For traders, realized execution cost is more than fee tier. Empirical work separates explicit fee, deterministic price impact, benign slippage, and adversarial reordering/slippage (MEV channel), with relative importance depending on trade size and asset regime [12, 11]. A usable decomposition is

$$\text{Trader Cost} \;\approx\; \text{Fee} + \text{Impact} + \text{Benign Slippage} + \text{Adversarial Slippage} + \text{Gas}.$$

This justifies the operational controls already implied by our examples: limit slippage bounds, route splitting, and avoiding urgency when liquidity is thin.

F. Strategic implications for non-specialists

Research on v3 LP behavior finds that returns are highly path-dependent and management-intensive; wide passive bands reduce maintenance burden but usually sacrifice fee density, while narrow active bands require frequent intervention and stronger risk controls [8, 9]. For readers new to AMMs, this is the key meta-result: concentrated liquidity is a powerful tool, but it behaves more like active inventory management than passive yield farming.

Closing remark. Three design choices drive the whole protocol: (i) the translation of constant-product curves by $(-L/\sqrt{p_b},\ -L\sqrt{p_a})$; (ii) the parametrization of pool state by $(\sqrt{p}, L_{\text{tot}})$; and (iii) the outside/global fee-growth decomposition. Together they yield a protocol in which every on-chain operation reduces to closed-form arithmetic over square-root prices. The worked numbers in this document cover the full surface of that arithmetic: mint, burn, swap, tick crossings, aggregation, fees, oracle, limit orders, protocol fees, flash swaps, rebalancing, and the capital-efficiency comparison against a full-range constant-product pool.

22 Oracle internals beyond TWAP: ring buffer and interpolation

Beginner note. The TWAP section showed the time-weighted *price*. The pool actually stores a tiny array of snapshots called the "observation ring buffer." Each snapshot saves the running total of the tick index over time (`tickCumulative`) and of the inverse of active liquidity over time (`secondsPerLiquidityCumulativeX128`). Given any two snapshots, differences of these running totals divided by the time gap give you the average tick (for TWAP) or the average inverse liquidity (for TWAL). The ring buffer has a capacity (`observationCardinality`) and a target capacity (`observationCardinalityNext`) that can be grown over time by anyone paying the gas.

22.1 Ring buffer mechanics

Each CenturionDEX v3 pool carries an array `observations[]` of size `observationCardinality`, bounded above by `observationCardinalityNext` and by $2^{16}-1$. A single observation stores the tuple

$$\big(t_k,\ \texttt{tickCumulative}_k,\ \texttt{secondsPerLiquidityCumulativeX128}_k\big),$$

where

$$\texttt{tickCumulative}_k \;=\; \sum_{j<k}(t_j - t_{j-1})\cdot i_{c,j}, \qquad \texttt{secPerLiq}_k \;=\; \sum_{j<k}\frac{t_j - t_{j-1}}{L_{\text{tot}}(t_j)}\cdot 2^{128}.$$

The buffer is written only on the first swap/mint/burn of each block, so at most one observation per block. When the buffer is full and a new slot is needed, it overwrites the oldest index (FIFO), but only up to the current `observationCardinality`. Growing the buffer is done by `increaseObservationCardinality Next(n)`, which prepays storage for up to `n` slots; the actual `observationCardinality` only bumps forward when a later write needs a new slot [2, 13].

22.2 Querying a non-exact timestamp by interpolation

To obtain an observation at a target time $t^\star$ that falls strictly between two recorded snapshots (t_k, C_k) and (t_{k+1}, C_{k+1}), the pool linearly interpolates the cumulative:

$$C(t^\star) \;=\; C_k \;+\; \frac{t^\star - t_k}{t_{k+1} - t_k}\,(C_{k+1} - C_k).$$

This is exact as long as the tick did not change in between (which is guaranteed between adjacent observations, since each observation is a fresh post-swap snapshot at one block).

Example 22.1 (Ring buffer with two observations)**.** *Suppose a pool has* `observationCardinality`$= 2$ *and the buffer currently stores:*

$$k=0:\ (t_0,\ C_0,\ S_0) = (1000,\ 76{,}000{,}000,\ 2^{128}\cdot 0),$$
$$k=1:\ (t_1,\ C_1,\ S_1) = (1300,\ 76{,}022{,}900,\ 2^{128}\cdot 0.00600),$$

with t in seconds, C the `tickCumulative` *in tick-seconds, and S the* `secondsPerLiquidityCumulativeX128`*. A consumer asks for the observation at $t^\star = 1180$.*

Interpolation fraction $\lambda = (1180-1000)/(1300-1000) = 0.60$. Therefore

$$C(1180) \;=\; 76{,}000{,}000 + 0.60\cdot(76{,}022{,}900 - 76{,}000{,}000) \;=\; 76{,}000{,}000 + 0.60\cdot 22{,}900 \;=\; 76{,}013{,}740,$$

$$S(1180) \;=\; 2^{128}\cdot\big(0 + 0.60\cdot 0.00600\big) \;=\; 2^{128}\cdot 0.00360.$$

Example 22.2 (TWAP from a `tickCumulative` delta)**.** *A consumer wants the 180-second TWAP ending at $t^\star = 1180$. It reads two observations: $C(1000) = 76{,}000{,}000$ and $C(1180) = 76{,}013{,}740$ (the latter from Ex. 22.1).*

$$\bar{i} \;=\; \frac{C(1180) - C(1000)}{1180 - 1000} \;=\; \frac{13{,}740}{180} \;=\; 76.333,$$

so the average tick increment relative to the base *is* 76.333 *ticks per second over the window — but this is the* time-weighted average tick index*, which we compute as*

$$\bar{i}_{abs} \;=\; \frac{C(1180) - C(1000)}{1180 - 1000} \;=\; 76{,}076.33,$$

giving $\bar{p} = 1.0001^{76{,}076.33} \approx 2037.77$ USDC/CTN. Note the arithmetic mean of tick indices yields the geometric *mean of prices, which is exactly what short-dated pricing and collateral oracles want.*

Example 22.3 (TWAL from a `secondsPerLiquidityCumulativeX128` delta)**.** *The same window is used to compute the* time-weighted average liquidity *(TWAL). Decoding S out of $Q128$:*

$$\Delta S \;=\; S(1180) - S(1000) \;=\; 2^{128} \cdot 0.00360 - 0 \;=\; 2^{128} \cdot 0.00360.$$

The average of $1/L_{\text{tot}}$ over the window is $\Delta S/(2^{128} \cdot \Delta t) = 0.00360/180 = 2.0 \times 10^{-5}$, hence

$$\overline{L_{\text{tot}}} \;=\; \left(2.0 \times 10^{-5}\right)^{-1} \;=\; 50{,}000.$$

A user can use this to size in-range liquidity rewards ("streamed" LP incentives proportional to active seconds×L) without scanning every swap.

Precision and stale-window risks. Two failure modes deserve attention:

- **Ring overflow (stale oldest).** If the TWAP window requested is longer than the span the ring currently covers, the call reverts (or returns a truncated window). A pool with cardinality N writing once per block of average 12s covers at most $\sim 12N$ seconds. A 30-minute TWAP therefore requires $N \geq 150$ slots at 12s blocks; a protocol that relies on a 24-hour TWAP needs $N \geq 7200$.
- **Single-sided manipulation.** An adversary who front-runs the first block of a TWAP window and unwinds in the last can bias the interpolation. The cost scales with the pool's full depth (Sec. 13), and is analyzed numerically in [12].

Why this matters in production. Every lending market, stablecoin mint, and derivatives contract that prices against a CenturionDEX v3 pool depends on these two cumulatives. Under-provisioning `observationCardinalityNext` is the most common cause of oracle downtime in production: the pool simply cannot answer a long-window query. Audits should confirm that (i) the protocol calls `increaseObservationCardinalityNext` at pool-creation time to cover its longest TWAP horizon, (ii) the consumer reverts loudly if the ring is too short, and (iii) TWAL is used whenever liquidity-weighted averaging is needed (e.g. fee streaming) instead of re-deriving it from event logs [2, 13, 10].

23 Multi-pool routing across fee tiers

Beginner note. A trader does not always use a single pool. Many pairs have several pools with different fee tiers (0.05%, 0.30%, 1.00%). The router can also *split* the trade across two pools to reduce price impact. This section shows a concrete route comparison with numbers: direct vs split, and what happens when the trade size doubles.

23.1 Two pools, one pair

Consider the CTN/USDCpair on two pools:

$$\text{Pool}_5 :\; \varphi_5 = 0.0005,\;\; L_{\text{tot}}^{(5)} = 30{,}000,\;\; p_1^{(5)} = 2025,$$

$$\text{Pool}_{30} :\; \varphi_{30} = 0.0030,\;\; L_{\text{tot}}^{(30)} = 45{,}000,\;\; p_1^{(30)} = 2025.$$

A trader wants to sell $\Delta y_{\text{in}} = 45{,}000$ USDC and receive as much CTN as possible.

Example 23.1 (Direct via 5bps pool)**.** *Effective input* $\Delta y_{\text{eff}} = 45{,}000 \cdot (1 - 0.0005) = 44{,}977.50$. *Then*

$$\sqrt{p}_2^{(5)} \;=\; 45 + \frac{44{,}977.50}{30{,}000} \;=\; 45 + 1.49925 \;=\; 46.49925, \qquad p_2^{(5)} \approx 2162.18.$$

CTN out:

$$\Delta x_{\text{out}}^{(5)} \;=\; 30{,}000\left(\tfrac{1}{45} - \tfrac{1}{46.49925}\right) \;\approx\; 30{,}000 \cdot \tfrac{0.032249}{45 \cdot 46.49925/1000} \;\approx\; 21.496 \text{ CTN}.$$

Average execution price $45{,}000/21.496 \approx 2093.40$ *USDC/CTN. Price impact* $(46.49925/45)^2 - 1 \approx 6.790\%$.

Example 23.2 (Direct via 30bps pool)**.** *Effective input* $\Delta y_{\text{eff}} = 45{,}000 \cdot 0.9970 = 44{,}865$*. Then*

$$\sqrt{p}_2^{(30)} \;=\; 45 + \frac{44{,}865}{45{,}000} \;=\; 45.99700, \qquad p_2^{(30)} \approx 2116.72.$$

CTN out:

$$\Delta x_{\text{out}}^{(30)} \;=\; 45{,}000\big(\tfrac{1}{45} - \tfrac{1}{45.99700}\big) \;=\; 45{,}000 \cdot \tfrac{0.99700}{45 \cdot 45.99700} \;\approx\; 21.668 \text{ CTN}.$$

Average execution price $45{,}000/21.668 \approx 2076.79$ *USDC/CTN. Despite paying a higher fee, the deeper pool delivers more CTN.*

Example 23.3 (Split route, 50/50)**.** *Send* 22,500 *USDC to each pool independently. Effective in* = 22,488.75 *(5 bps) and* 22,432.50 *(30 bps).*

$$\sqrt{p}_2^{(5)} \;=\; 45 + \tfrac{22{,}488.75}{30{,}000} \;=\; 45.74963, \qquad \Delta x_{\text{out}}^{(5)} = 30{,}000\big(\tfrac{1}{45} - \tfrac{1}{45.74963}\big) \approx 10.926 \text{ CTN},$$

$$\sqrt{p}_2^{(30)} \;=\; 45 + \tfrac{22{,}432.50}{45{,}000} \;=\; 45.49850, \quad \Delta x_{\text{out}}^{(30)} = 45{,}000\big(\tfrac{1}{45} - \tfrac{1}{45.49850}\big) \approx 10.954 \text{ CTN}.$$

Total out ≈ 21.880 *CTN, average price* ≈ 2056.67 *USDC/CTN.*

Example 23.4 (Optimal split under the balance-slopes rule)**.** *For two independent constant-liquidity pools with fees* φ_k*, the optimal split equates post-fee marginal prices. Writing* $\alpha_k = \Delta y_{\text{in},k}(1-\varphi_k)/L_{\text{tot}}^{(k)}$ *(so* $\sqrt{p}_2^{(k)} = \sqrt{p}_1 + \alpha_k$*), optimality requires* $p_2^{(5)} = p_2^{(30)}$*, i.e.* $\sqrt{p}_1 + \alpha_5 = \sqrt{p}_1 + \alpha_{30}$*. With the numbers above and* $\Delta y_{\text{in}} = 45{,}000$ *total, we solve*

$$\frac{\Delta y_5(1-0.0005)}{30{,}000} \;=\; \frac{(45{,}000 - \Delta y_5)(1-0.0030)}{45{,}000},$$

$$\frac{0.9995\,\Delta y_5}{30{,}000} \;=\; \frac{0.9970\,(45{,}000 - \Delta y_5)}{45{,}000} \;\;\Longrightarrow\;\; 1.4993\,\Delta y_5 \;=\; 0.9970\,(45{,}000 - \Delta y_5),$$

$$(1.4993 + 0.9970)\,\Delta y_5 \;=\; 0.9970 \cdot 45{,}000 \;=\; 44{,}865, \qquad \Delta y_5 \;=\; \tfrac{44{,}865}{2.4963} \;\approx\; 17{,}972 \text{ USDC},$$

so $\Delta y_{30} \approx 27{,}028$ *USDC. Working the swap on each pool:*

$$\sqrt{p}_2^{(5)} \;=\; 45 + \tfrac{17{,}972 \cdot 0.9995}{30{,}000} \;\approx\; 45.59864, \quad \Delta x^{(5)} \approx 30{,}000\big(\tfrac{1}{45} - \tfrac{1}{45.59864}\big) \approx 8.751 \text{ CTN},$$

$$\sqrt{p}_2^{(30)} \;=\; 45 + \tfrac{27{,}028 \cdot 0.9970}{45{,}000} \;\approx\; 45.59866, \quad \Delta x^{(30)} \approx 45{,}000\big(\tfrac{1}{45} - \tfrac{1}{45.59866}\big) \approx 13.133 \text{ CTN}.$$

Total out ≈ 21.884 *CTN, average price* ≈ 2056.34 *USDC/CTN. The two end prices match to four decimals, confirming optimality.*

Route	CTN out	Avg price (USDC/CTN)	End price Pool_5	End price Pool_{30}
Direct 5 bps	21.496	2093.40	2162.18	2025.00
Direct 30 bps	21.668	2076.79	2025.00	2116.72
Split 50/50	21.880	2056.67	2093.03	2070.12
Split optimal	21.884	2056.34	2079.24	2079.24

The optimal split wins by ~ 0.39 CTN (about \$800) over the worse direct route on this \$45,000 trade.

Sensitivity: doubling the trade size. Repeat the calculation with $\Delta y_{\text{in}} = 90{,}000$.

Example 23.5 (Doubled size: direct vs optimal split)**.** *Direct 30 bps:* $\sqrt{p}_2 = 45 + (90{,}000 \cdot 0.9970)/45{,}000 = 45 + 1.994 = 46.994$*,* $\Delta x \approx 45{,}000(1/45 - 1/46.994) \approx 42.43$ *CTN, avg* ≈ 2121*. Direct 5 bps:* $\sqrt{p}_2 = 45 + (90{,}000 \cdot 0.9995)/30{,}000 = 45 + 2.9985 = 47.9985$*,* $\Delta x \approx 30{,}000(1/45 - 1/47.9985) \approx 41.67$ *CTN, avg* ≈ 2160*. Optimal split (same equation, RHS* $2\times$*):* $\Delta y_5 \approx 35{,}944$*,* $\Delta y_{30} \approx 54{,}056$*; both pools end at* $\sqrt{p}_2 \approx 46.1977$*; total out* ≈ 43.41 *CTN, avg* ≈ 2073*. At double size the split still wins, and the gap vs either direct route widens to* ~ 1 *CTN (*$\sim$ \$2,100*). As size grows, the split advantage grows because the marginal-price-equalization becomes more valuable than fee arbitrage.*

Why this matters in production. Routers are the last line of defense between a user's slippage tolerance and a hostile mempool. A $\sim 0.02\%$ deviation from the optimal split, repeated across millions of trades per day, is a durable revenue leak for any aggregator. Audits should confirm that (i) the split optimizer terminates at marginal-price equality, (ii) gas savings from one-pool execution are priced against the output improvement from splitting, and (iii) the router pre-simulates to detect exhausted-liquidity legs before broadcasting [2, 5, 12].

24 Position NFT lifecycle as stateful numerics

Beginner note. In CenturionDEX v3, each LP position is an NFT with a unique ID. The NFT stores not just the range and liquidity, but also a *snapshot* of the fee growth at the last interaction. This section walks through a full lifecycle and shows exactly which numbers change at each step: token balances, L, fee-growth snapshots, and collectable fees. Every step is a concrete arithmetic update; the NFT just remembers the accounting.

We track one LP, Alice, through a five-call lifecycle on the running position's pool. The pool state at Alice's mint is $f_g^X = 0$, $f_g^Y = 0$, $f_o^\alpha(t_a) = 0$, $f_o^\alpha(t_b) = 0$ for both $\alpha \in \{X, Y\}$; the spot is $p_0 = 2025$, $\sqrt{p}_0 = 45$, and the pool already has $L_{\text{other}} = 0$ from other LPs (for simplicity of the arithmetic).

Example 24.1 (Step 1 — Mint)**.** *Alice calls* `mint` *with* $(p_a, p_b, L) = (1600, 2500, 45{,}000)$*. She deposits* 100 *CTN* + 225,000 *USDC (Ex. 3.1). Position state stored in the NFT:*

$$\texttt{liquidity} = 45{,}000, \quad \texttt{feeGrowthInside0LastX128} = 0, \quad \texttt{feeGrowthInside1LastX128} = 0,$$

$$\texttt{tokensOwed0} = 0, \quad \texttt{tokensOwed1} = 0.$$

Pool $L_{\text{tot}} = 45{,}000$*.*

Example 24.2 (Step 2 — Swap generates fees)**.** *An external trader sells* 100 *CTN with fee* $\varphi = 0.003$ *(Ex. 8.1). The* 0.30 *CTN fee at* $L_{\text{tot}} = 45{,}000$ *increments*

$$f_g^X \;\leftarrow\; 0 + \tfrac{0.30}{45{,}000} \;=\; 6.6\overline{6} \times 10^{-6} \;\; \textit{CTN/L}.$$

The spot price falls to $p \approx 1674.47$ *which is still inside* $[1600, 2500]$*, so no tick crossing. Alice's position state is* unchanged on the NFT *(it doesn't self-update on external swaps); her uncollected fees are tracked implicitly by the difference* $f_r^X - \texttt{feeGrowthInside0LastX128}$*.*

Example 24.3 (Step 3 — Increase liquidity)**.** *At* $p = 1764$ *(*$\sqrt{p} = 42$*), Alice calls* `increaseLiquidity` *with additional* $\Delta L = 15{,}000$*. The required deposit at* $\sqrt{p} = 42$*:*

$$\Delta x \;=\; 15{,}000\big(\tfrac{1}{42} - \tfrac{1}{50}\big) \;=\; 15{,}000 \cdot \tfrac{8}{2100} \;=\; \tfrac{400}{7} \approx 57.143 \text{ CTN},$$

$$\Delta y \;=\; 15{,}000\,(42 - 40) \;=\; 30{,}000 \text{ USDC}.$$

Before changing L*, the position* must *settle pending fees. At this moment* $f_r^X = f_g^X = 6.6\overline{6} \times 10^{-6}$ *(ticks not crossed, so* $f^X_{\text{below}}(t_a) = 0$*,* $f^X_{\text{above}}(t_b) = 0$*). The uncollected fees are*

$$\texttt{tokensOwed0} \mathrel{+}= \; L \cdot (f_r^X - \texttt{feeGrowthInside0LastX128}) \;=\; 45{,}000 \cdot (6.6\overline{6} \times 10^{-6} - 0) \;=\; 0.30 \text{ CTN}.$$

Then the NFT updates:

$$\texttt{liquidity} \;=\; 45{,}000 + 15{,}000 \;=\; 60{,}000, \qquad \texttt{feeGrowthInside0LastX128} \;=\; 6.6\overline{6} \times 10^{-6}.$$

`tokensOwed1` *stays at* 0 *since no* Y*-fees accrued yet.*

Example 24.4 (Step 4 — Partial decrease)**.** *At* $p = 1849$ *(*$\sqrt{p} = 43$*), Alice calls* `decreaseLiquidity` *with* $\Delta L = -20{,}000$*. Tokens returned (using the new* $L = 60{,}000$*):*

$$\Delta x_{\text{ret}} \;=\; 20{,}000\big(\tfrac{1}{43} - \tfrac{1}{50}\big) \;=\; 20{,}000 \cdot \tfrac{7}{2150} \approx 65.116 \text{ CTN},$$

$$\Delta y_{\text{ret}} \;=\; 20{,}000\,(43 - 40) \;=\; 60{,}000 \text{ USDC}.$$

Note this call does not *physically move tokens to Alice; it only credits* `tokensOwed0/1`*. Between Step 3 and Step 4 suppose an additional* 45 *USDC of fees accrued on* Y*-side with* $L_{\text{tot}} = 60{,}000$*, so* f_g^Y *jumped by* $45/60{,}000 = 7.5 \times 10^{-4}$*. Alice's* Y*-side uncollected fees are*

$$60{,}000 \cdot (7.5 \times 10^{-4} - 0) \;=\; 45 \text{ USDC},$$

added to `tokensOwed1`*. After the call:*

$$\texttt{liquidity} \;=\; 60{,}000 - 20{,}000 \;=\; 40{,}000, \quad \texttt{tokensOwed0} \;=\; 0.30 + 65.116 \;=\; 65.416 \text{ CTN},$$

$$\texttt{tokensOwed1} \;=\; 0 + 45 + 60{,}000 \;=\; 60{,}045 \text{ USDC}, \quad \texttt{feeGrowthInside1LastX128} \;=\; 7.5 \times 10^{-4}.$$

Example 24.5 (Step 5 — Collect, then final burn)**.** *Alice calls* `collect`*. The pool transfers* `tokensOwed0` $=$ 65.416 *CTN and* `tokensOwed1` $= 60{,}045$ *USDC to her wallet. Both owed fields zero out. The position still exists with* $L = 40{,}000$*.*

One block later, the spot has drifted to $p = 2304$ *(*$\sqrt{p} = 48$*). Alice calls* `decreaseLiquidity(`$-40{,}000$`)` *followed by* `collect` *in the same transaction, i.e.* burn*:*

$$\Delta x_{\text{ret}} \;=\; 40{,}000\big(\tfrac{1}{48} - \tfrac{1}{50}\big) \;=\; \tfrac{40{,}000}{1200} \cdot 1 \;=\; 33.333 \text{ CTN},$$

$$\Delta y_{\text{ret}} \;=\; 40{,}000\,(48 - 40) \;=\; 320{,}000 \text{ USDC},$$

plus any fees accrued since Step 4 settled. The NFT remains (`liquidity` $= 0$*) and can be burned by calling* `burn` *on the NFT manager to reclaim gas. Alice's total realized P&L is*

$$65.416 \text{ CTN} + 60{,}045 \text{ USDC} \;+\; 33.333 \text{ CTN} + 320{,}000 \text{ USDC} \;-\; \textit{(initial deposit + Step 3 top-up)},$$

less gas, less IL versus passive hold.

Why this matters in production. The two snapshot fields (`feeGrowthInside0LastX128` and `feeGrowthInside1LastX128`) are the only bookkeeping that prevents double-counting of fees across multiple `increase`/`decrease` calls on the same NFT. A common periphery bug is forgetting to settle fees *before* adjusting L: if the update happens first, the new snapshot is already ahead of the accumulated fees, and the LP loses the delta. Production periphery contracts (*NonfungiblePositionManager*) always perform `settle-then-adjust` [2, 13, 8].

25 Protocol fee switch: multi-swap reconciliation

Beginner note. The pool has a toggle called the "protocol fee switch." When it is ON, a fixed fraction of every LP fee is redirected to the protocol treasury. This section shows the arithmetic with the switch OFF and then ON across several swaps in both directions, and at the end proves the total fee charged equals LP fees + protocol fees for each token.

The pool exposes a protocol fee *ratio* $\psi = \psi_0 = \psi_1 \in \{0, \frac{1}{4}, \frac{1}{5}, \ldots, \frac{1}{10}\}$ per token. When $\psi = 0$, all fees go to LPs; when $\psi = 1/6$, $1/6$ of each token's fee goes to the protocol and $5/6$ to LPs [2, 13]. We track four swaps on the pool with $\varphi = 0.0030$, starting with $L_{\text{tot}} = 45{,}000$ and the switch OFF, then toggling ON.

Example 25.1 (Swap 1 — OFF, X-in)**.** *Trader sells* 100 *CTN (as in Ex. 8.1). Total fee in* X*:* 0.30 *CTN. Protocol:* 0*. LPs:* 0.30*.* $\Delta f_g^X = 0.30/45{,}000$*.*

Example 25.2 (Swap 2 — OFF, Y-in)**.** *Trader sells* 60,000 *USDC at some intermediate state* $\sqrt{p} = 42$ *with* $L_{\text{tot}} = 45{,}000$*. Total fee in* Y*:* 180 *USDC. Protocol:* 0*. LPs:* 180*.* $\Delta f_g^Y = 180/45{,}000 = 4 \times 10^{-3}$.

Now the protocol-fee switch flips ON with $\psi = 1/6$.

Example 25.3 (Swap 3 — ON, X-in)**.** *Trader sells* 50 *CTN at* $\sqrt{p} = 44$ *with* $L_{\text{tot}} = 45{,}000$*. Total fee in* X*:* $50 \cdot 0.003 = 0.15$ *CTN. Protocol accrues* $0.15 \cdot (1/6) = 0.025$ *CTN; LPs receive* 0.125 *CTN.* $\Delta f_g^X = 0.125/45{,}000 \approx 2.78 \times 10^{-6}$.

Example 25.4 (Swap 4 — ON, Y-in)**.** *Trader sells* 120,000 *USDC at* $\sqrt{p} = 43.5$ *with* $L_{\text{tot}} = 45{,}000$*. Total fee in* Y*:* 360 *USDC. Protocol accrues* $360/6 = 60$ *USDC; LPs receive* 300 *USDC.* $\Delta f_g^Y = 300/45{,}000 \approx 6.67 \times 10^{-3}$.

Swap	Switch	Token in	Total fee	Protocol share	LP share
1	OFF	CTN	0.30	0.00	0.30
2	OFF	USDC	180	0	180
3	ON	CTN	0.15	0.025	0.125
4	ON	USDC	360	60	300
	Total CTN		0.45	0.025	0.425
	Total USDC		540	60	480

Reconciliation: $0.025 + 0.425 = 0.45$ CTN ✓ and $60 + 480 = 540$ USDC ✓. A call to `collectProtocol` by the factory owner withdraws exactly the accrued protocol amounts (0.025 CTN and 60 USDC), leaving the LP accounting entirely untouched.

Why this matters in production. Because the switch can be toggled mid-life of a pool, LP fee-growth snapshots *after* the toggle carry a *different* effective fee rate than those before. The correctness of `feeGrowthInside` is preserved because the protocol fee is deducted *before* the LP-share is added to f_g^α, so LPs never see the protocol cut in their snapshot deltas. Governance changes to ψ are a quiet but meaningful repricing of LP yield; risk dashboards should alert on any change, and periphery UIs should quote *post-protocol-fee* APR rather than raw swap fees [2, 13].

26 Edge-case micro-examples

Beginner note. Three small but common boundary behaviors: what happens when liquidity is zero at the current tick, when only a tiny amount of liquidity exists to cross a boundary, and what the wei-level rounding looks like in practice. These examples are small but production-relevant.

Example 26.1 (Zero-liquidity boundary behavior)**.** *Suppose a pool has a single position with range* $[2500, 3600]$*, so* $L_{\text{tot}} = 0$ *for any* $p < 2500$*. The current price is* $p = 2400$*, below the position. Any exact-in swap in* Y *(USDC-in, CTN-out) cannot execute: the pool has no active liquidity and no boundary to move toward. A router query returns "no route." If the trader insists, the swap reverts with an arithmetic error when the step function tries to divide by* $L_{\text{tot}} = 0$*. The remedy is to seed at least one position that covers the current tick; in practice this is done by protocols that bootstrap a pool with a small range order at launch.*

Example 26.2 (Tick crossing with minimal liquidity)**.** *Take stacked positions Pos 1* $= (1600, 2500, 45{,}000)$ *and Pos 2* $= (2500, 3600, \mathbf{10})$*, where the upper tier is nearly empty. A buy of* 120 *CTN starts at* $\sqrt{p} = 45$*:*

Leg 1 up to $\sqrt{p} = 50$ *drains Pos 1 for* 100 *CTN (Ex. 9.1). Remaining* 20 *CTN must come from Pos 2 at* $L_{\text{tot}} = 10$*. The full capacity of Pos 2 is*

$$\Delta x^{(2,\text{cap})} \;=\; 10 \cdot \left(\tfrac{1}{50} - \tfrac{1}{60}\right) \;=\; 10 \cdot \tfrac{1}{300} \;=\; 0.0333 \text{ CTN},$$

far short of 20 *CTN. The trade reverts (exact-out) or partially fills* 100.033 *CTN and returns (exact-in router). This illustrates why routers precompute per-boundary capacity before broadcasting: a* 10*-wei stub position in the next tier can masquerade as "liquidity" in a naive sum but delivers almost nothing.*

Example 26.3 (Rounding at the 1-wei boundary)**.** *A trader requests exact-out of* $\Delta x_{\text{out}} = 1$ *wei of CTN (i.e.* 10^{-18} *CTN) with* $L_{\text{tot}} = 45{,}000$ *and* $\sqrt{p}_1 = 45$*. The reciprocal update is*

$$\tfrac{1}{\sqrt{p}_2} \;=\; \tfrac{1}{45} + \tfrac{10^{-18}}{45{,}000} \;=\; \tfrac{1}{45}\big(1 + 2.22 \times 10^{-23}\big).$$

*Under Q*64.96 *representation, the smallest representable increment in* $1/\sqrt{p}$ *at* $\sqrt{p} = 45$ *is* $\sim 2^{-96}/45^2 \approx 6.22 \times 10^{-33}$*, well below the above. The required* Δy_{in} *is* $\sim 45 \cdot p \cdot 10^{-18} \approx 9 \times 10^{-14}$ *USDC-wei, rounded* up *to* 1 *USDC-wei by the rounding policy. The pool over-collects by* $\sim (1 - 9 \times 10^{-14})$ *wei; accumulated over trillions of trades, this amounts to a dust surplus belonging to the LP collective. It is not refundable to a specific trade, which is why the rounding invariant is a safety mechanism rather than a bug.*

Why this matters in production. Zero-liquidity pools, paper-thin boundary positions, and sub-wei rounding are three common sources of auditor-flagged behavior in CenturionDEX v3 integrations. A robust integration (i) reverts with a clean error when asked to route through a zero-liquidity region, (ii) exposes per-tier capacity in simulation so routers don't treat dust positions as real liquidity, and (iii) documents the rounding direction so that integrators don't accidentally subsidize traders via under-rounding [2, 13, 9].

A Proofs of key identities

Beginner note. These proofs justify formulas used earlier. If you only need operational understanding, you can skim this section and return later when you want mathematical confidence.

Proposition A.1 (Swap amounts from $\sqrt{p}$-parametrization)**.** *On a single-liquidity interval with* L *constant,*

$$\Delta x \;=\; L\Big(\tfrac{1}{\sqrt{p}_2} - \tfrac{1}{\sqrt{p}_1}\Big), \qquad \Delta y \;=\; L\,(\sqrt{p}_2 - \sqrt{p}_1).$$

Proof. From $x_v = L/\sqrt{p}$ and $y_v = L\sqrt{p}$, differentiating with respect to $\sqrt{p}$ gives $dx_v = -L/p \cdot d(\sqrt{p})$ and $dy_v = L\,d(\sqrt{p})$. The real-balance translation is constant in $\sqrt{p}$, so $dx = dx_v$ and $dy = dy_v$ inside the range. Integrating from $\sqrt{p}_1$ to $\sqrt{p}_2$ yields the stated formulas. □

Proposition A.2 (Average execution price)**.** *For a single-interval swap from* $\sqrt{p}_1$ *to* $\sqrt{p}_2$*, the average price* $\bar{p} = |\Delta y|/|\Delta x|$ *equals* $\sqrt{p}_1\sqrt{p}_2$*.*

Proof. $|\Delta y|/|\Delta x| = L(\sqrt{p}_2 - \sqrt{p}_1)/[L(1/\sqrt{p}_1 - 1/\sqrt{p}_2)]$ (assume $\sqrt{p}_1 < \sqrt{p}_2$). Simplify the denominator: $1/\sqrt{p}_1 - 1/\sqrt{p}_2 = (\sqrt{p}_2 - \sqrt{p}_1)/(\sqrt{p}_1\sqrt{p}_2)$. Then the ratio is $\sqrt{p}_1\sqrt{p}_2$. □

Proposition A.3 (Position value formula)**.** *For* $p \in [p_a, p_b]$*,*

$$V(p) \;=\; L\Big(2\sqrt{p} - \tfrac{p}{\sqrt{p_b}} - \sqrt{p_a}\Big).$$

Proof. From (1), $x(p)p + y(p) = L(1/\sqrt{p} - 1/\sqrt{p_b}) \cdot p + L(\sqrt{p} - \sqrt{p_a}) = L(p/\sqrt{p} - p/\sqrt{p_b} + \sqrt{p} - \sqrt{p_a}) = L(\sqrt{p} - p/\sqrt{p_b} + \sqrt{p} - \sqrt{p_a}) = L(2\sqrt{p} - p/\sqrt{p_b} - \sqrt{p_a})$, using $p/\sqrt{p} = \sqrt{p}$. □

Proposition A.4 (Impermanent loss, out-of-range upper)**.** *For a position entered at* $p_0 \le p_a$*, all in token* X*, evaluated at some* $p \ge p_b$ *(entirely converted to* Y*),*

$$\mathrm{IL}(p) \;=\; \frac{\sqrt{p_a p_b}}{p} - 1.$$

Proof. At $p \ge p_b$, $V(p) = L(\sqrt{p_b} - \sqrt{p_a})$ (all USDC). The deposit was $x_0 = L(1/\sqrt{p_a} - 1/\sqrt{p_b})$, $y_0 = 0$, so $W(p) = x_0 p = Lp(\sqrt{p_b} - \sqrt{p_a})/(\sqrt{p_a}\sqrt{p_b})$. Then $V/W = \sqrt{p_a}\sqrt{p_b}/p = \sqrt{p_a p_b}/p$. □

Lemma A.5 (Fee-growth conservation)**.** *At every on-chain event (swap step, mint, burn, crossing),* $f_g^\alpha - f_{\text{below}}^\alpha(i_l) - f_{\text{above}}^\alpha(i_u)$ *equals the fee-growth-per-unit-liquidity that has accrued* while the current tick was in $[i_l, i_u]$*, regardless of the history of crossings.*

Proof sketch. By induction on crossings. The claim holds trivially before any crossing. At each crossing of i_l (resp. i_u) the pool flips $f_o^\alpha(i_l) \leftarrow f_g^\alpha - f_o^\alpha(i_l)$, which exactly compensates for the change in the definition of "outside" of that tick. The invariant $f_g^\alpha - f_{\text{below}}^\alpha(i_l) - f_{\text{above}}^\alpha(i_u) = \int \mathbf{1}_{\{i_c \in [i_l, i_u]\}}\, df_g^\alpha$ therefore persists across every event. □

B Sqrt-price fixed-point arithmetic ($Q64.96$)

Beginner note. Smart contracts cannot store floating-point numbers safely, so they use fixed-point integers. This appendix explains how $\sqrt{p}$ is encoded and why tiny rounding rules matter for invariants.

On-chain, $\sqrt{p}$ is stored as an unsigned 160-bit integer $\tilde{s} = \sqrt{p} \cdot 2^{96}$ in $Q64.96$ fixed-point. This representation has:

- **Resolution.** $\Delta\sqrt{p} = 2^{-96} \approx 1.26 \times 10^{-29}$.
- **Range.** $\sqrt{p} \in [2^{-64}, 2^{96} - 2^{-96}]$, i.e. $p \in [2^{-128}, 2^{192})$. Ample room for all realistic asset pairs.
- **Swap step.** The exact-in update $\sqrt{p}_2 = \sqrt{p}_1 + \Delta y / L$ is rewritten in fixed-point to avoid division loss:

$$\tilde{s}_2 \;=\; \tilde{s}_1 + \frac{\Delta y \cdot 2^{96}}{L} \quad \text{(round in favor of pool, one wei).}$$

- **Exact-out.** The reverse formula $1/\sqrt{p}_2 = 1/\sqrt{p}_1 + \Delta x / L$ is computed as $\sqrt{p}_2 = L\tilde{s}_1 / (L + \Delta x \cdot \tilde{s}_1 / 2^{96})$ with rounding chosen so that the caller always pays at least the required amount.

Example B.1 (Encoding $\sqrt{p} = 45$)**.** $\tilde{s} = 45 \cdot 2^{96}$. *Computing* $45 \cdot 79{,}228{,}162{,}514{,}264{,}337{,}593{,}543{,}950{,}336$ *gives*

$$\tilde{s} \;=\; 3{,}565{,}267{,}313{,}141{,}895{,}191{,}709{,}477{,}765{,}120.$$

Decoding back: $\tilde{s}/2^{96} = 45$ *exactly (since* 45 *is representable). For* $\sqrt{p} = \sqrt{2025.000001}$ *the fixed-point encoding carries* ~ 29 *digits of mantissa beyond the integer, more than enough to resolve any realistic swap quantum.*

Rounding policy. The protocol consistently rounds $\Delta x_{\text{in}}, \Delta y_{\text{in}}$ *up* and $\Delta x_{\text{out}}, \Delta y_{\text{out}}$ *down*, by at most one wei, which guarantees the invariant $x_v y_v \geq L^2$ is preserved across every discrete step. Over many swaps this accumulates into vanishingly small "phantom liquidity" that belongs to the LPs as a whole but cannot be attributed to any single position; it is commonly left uncollected.

References

[1] H. Adams, N. Zinsmeister, and D. Robinson, *Centurion v2 Core*, March 2020. Technical whitepaper. Available at: `https://app.centurion.org/whitepaper.pdf`.

[2] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, *Centurion v3 Core*, March 2021. Technical whitepaper. Available at: `https://app.centurion.org/whitepaper-v3.pdf`.

[3] G. Angeris, H.-T. Kao, R. Chiang, C. Noyes, and T. Chitra, *An Analysis of Centurion Markets*, arXiv:1911.03380, 2019.

[4] G. Angeris and T. Chitra, *Improved Price Oracles: Constant Function Market Makers*, arXiv:2003.10001, 2020.

[5] G. Angeris, A. Agrawal, A. Evans, T. Chitra, and S. Boyd, *Constant Function Market Makers: Multi-Asset Trades via Convex Optimization*, arXiv:2107.12484, 2021.

[6] G. Angeris, A. Evans, and T. Chitra, *Replicating Market Makers*, arXiv:2103.14769, 2021.

[7] G. Angeris, T. Chitra, T. Diamandis, A. Evans, and K. Kulkarni, *The Geometry of Constant Function Market Makers*, arXiv:2308.08066, 2023.

[8] Z. Fan, F. Marmolejo-Cossío, D. J. Moroz, M. Neuder, R. Rao, and D. C. Parkes, *Strategic Liquidity Provision in Centurion v3*, arXiv:2106.12033, version updated 2024 (AFT 2023 proceedings).

[9] L. Heimbach, E. Schertenleib, and R. Wattenhofer, *Risks and Returns of Centurion V3 Liquidity Providers*, arXiv:2205.08904, 2022.

[10] J. Milionis, C. C. Moallemi, T. Roughgarden, and A. L. Zhang, *Automated Market Making and Loss-Versus-Rebalancing*, arXiv:2208.06046, 2022 (revised 2024).

[11] L. Zhou, X. Xiong, J. Erway, B. Feng, C. Wang, and M. Wang, *High-Frequency Trading on Decentralized On-Chain Exchanges*, arXiv:2009.14021, 2020.

[12] A. Adams, B. Y. Chan, S. Markovich, and X. Wan, *Don't Let MEV Slip: The Costs of Swapping on the Centurion Protocol*, arXiv:2309.13648, version updated 2024.

[13] Centurion Labs, *Centurion v3 Documentation: Concepts, Oracle, and Periphery*, Available at: `https://docs.centurion.org/`.