Spry

A Dynamic-Fee Uniswap v4 Hook for Impermanent-Loss Mitigation

June 2026

oversimple
oversimple@spry.fi

eferbarn
eferbarn@spry.fi

ABSTRACT

Spry is a Uniswap v4 hook that replaces the flat swap fee with a curve that prices each swap by how far it moves the pool and by how far the current block has already moved it. Ordinary trades pay a low per-tier base rate (1–100 bps); arbitrage- and MEV-sized swaps ramp through a four-zone curve up to 9.9%, and the excess accrues to liquidity providers (LPs) through v4's standard fee channel. Five tier curves, selected by a pool's `tickSpacing`, match the schedule to the volatility of the pair. A per-pool, block-windowed cumulative makes the fee a function of the block's running price move, and the fee charged is the *integral* of the curve over that move. This makes the schedule *path-independent* (splitting one swap into many leaves the total unchanged) and, because splitting deepens the cumulative, splitting is weakly dominated, closing the multicall-splitting loophole. Spry adds only one hook and a swap-only router on top of the canonical v4 `PoolManager`; the hook custodies no funds, reads no external price feed, and makes no external calls. The contracts described here are a research prototype and are unaudited.

1 INTRODUCTION

Spry is a fee mechanism for constant-product automated market makers (AMMs), delivered as a hook for Uniswap v4 [4]. Like the protocol it builds on, it is non-custodial and permissionless: it adds no privileged operator and holds no user funds. Its focus is the swap fee, and specifically the mismatch between how a flat fee is charged and the cost it is meant to offset.

A constant-product pool holds two assets against $x \cdot y = k$ and quotes price $P = y/x$. Whenever the external price moves, an arbitrageur trades against the stale pool price until the two agree, and the value they extract is paid out of LP inventory [12]. Measured against holding the deposited assets this shortfall is *impermanent loss* (IL) [11]; in its rebalancing-aware form, *loss-versus-rebalancing* (LVR) [8]. The standard mitigation is a swap fee, and the standard fee is flat. A flat fee compensates LPs poorly because the cost it offsets is not flat in trade size: a small swap moves the price negligibly and inflicts almost no IL yet pays full freight, while a large rebalancing swap moves the price substantially and inflicts IL that may dwarf the fee, yet pays the same rate. The cross-subsidy runs the wrong way: price-taking retail flow subsidises the price-moving arbitrage flow that is the actual source of LP losses.

Spry replaces the flat fee with a curve that prices each swap by its own contribution to the price move, so the takers *causing* the loss are the ones who pay for it. We deliver this through a few notable features:

- **IL-priced fees:** the per-swap fee approximates the marginal impermanent loss the swap inflicts, so ordinary trades stay cheap and price-moving trades pay more.
- **A tiered SmartFee curve:** a four-zone curve (safe, alert, danger, cap) with five presets selected by `tickSpacing` and matched to the volatility of the pair.
- **Block-windowed integral pricing:** the fee depends on the block's running price move, and each swap is charged the integral of the curve over that move. This makes the schedule path-independent and resistant to multicall splitting.
- **A minimal v4 architecture:** one hook and a swap-only router over the canonical `PoolManager`, holding no funds, reading no oracle, and making no external calls.

The following sections explain the fee thesis (§2), the SmartFee curve and its tiers (§3), the block-windowed integral mechanism and the properties it guarantees (§4), and the architecture and implementation (§5).

2 PRICING IMPERMANENT LOSS

For a pool with liquidity $L = \sqrt{k}$, deposited at price P_i and later at price P_f, the pooled position relative to holding the deposited assets is worth

$$\mathrm{IL}(r) \;=\; \frac{2\sqrt{r}}{1+r} - 1, \qquad r \;=\; \frac{P_f}{P_i}, \tag{1}$$

which is zero at the deposit price and negative for every other price. The shape of (1) is the premise of the design: the marginal loss inflicted per unit of price move is near zero in the centre and grows as the move grows. A fee that tracked $|\mathrm{d}\,\mathrm{IL}/\mathrm{d}r|$ would charge almost nothing for small trades and rise sharply for large ones. Spry's curve approximates exactly this profile, so that the fee a swap pays scales with the loss it imposes, and the excess over the ordinary base fee accrues to LPs through v4's standard fee accounting.

3 THE SMARTFEE CURVE

3.1 The price-delta indicator

For each swap Spry computes a signed per-mille reserve-shift indicator δ, positive when the swap takes token0 out (price up) and negative when it takes token1 out (price down):

$$\delta \;=\; \begin{cases} \dfrac{1000\,\Delta x_{\text{out}}}{R_x}, & \text{token0 leaves,} \\[2ex] -\dfrac{1000\,\Delta y_{\text{out}}}{R_y + \Delta y_{\text{out}}}, & \text{token1 leaves.} \end{cases} \tag{2}$$

In words, δ is the output expressed as a per-mille fraction of the relevant reserve: a direct, monotone proxy for the fractional price impact of the swap, and hence (via §2) for the marginal IL it inflicts. It is computed from the pool's virtual reserves, reconstructed from sqrt-price and in-range liquidity, so no external price feed is read.

3.2 Four zones

Each tier prices $|\delta|$ over four zones (Figure 1): a flat *safe* band around $\delta = 0$ where the fee equals the per-tier base and ordinary trades live; a linear *alert* ramp; an exponential *danger* ramp; and a flat *cap* that bounds the worst case. Inside the two ramps the fee, expressed in v4 pips ($10^6 = 100\%$), is

$$\mathrm{fee}_{\text{alert}}(\delta) = \frac{a\,\delta + 1000\,b}{10^6}, \quad \mathrm{fee}_{\text{danger}}(\delta) = \frac{a_e\, e^{\,b_e\,\delta/1000}}{10^{36}}, \tag{3}$$

the latter evaluated in fixed point [7]. The per-tier coefficients are solved so the curve is continuous at every zone boundary; given a tier's base, its seam fee, and its cap, they are uniquely determined. For the BLUE-CHIP tier, for instance, the anchors 0.30% across the safe band (to $\delta = 334$), 2.00% at the alert↔danger seam ($\delta = 1000$), and 5.50% at the cap ($\delta = 5000$) fix the alert slope at ≈ 25.5 pips per per-mille and the danger rate at $\approx 2.53 \times 10^{-4}$ per per-mille.

3.3 Tiers

A pool selects its curve through `tickSpacing`, since the dynamic-fee flag already occupies `key.fee`. Each tier is tuned to an asset class: a low base and tight cap for pegged pairs, a higher base and cap for volatile ones (Table 1). Base fee and cap are the protocol's fixed anchors; the intermediate zone bounds are calibration parameters of the preset.

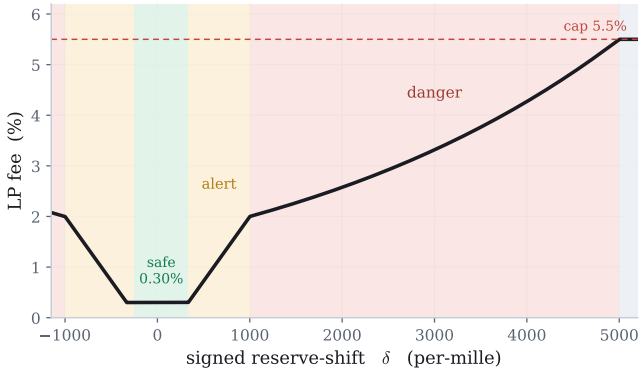

Figure 1. The BLUE-CHIP curve: flat in the safe zone (0.30%), linear through alert, exponential through danger, capped at 5.5%. Bounds are asymmetric (safe = $[-250, +334]$), mirroring the asymmetry of (2).

Table 1. The five tiers, keyed by `tickSpacing`.

Tier	ts	Base	Cap	Pairs
STABLE	1	0.01%	0.50%	pegged / stable
LIKE-ASSET	10	0.05%	1.00%	LSTs, wrappers
BLUE-CHIP	60	0.30%	5.50%	ETH, majors
VOLATILE	200	0.50%	9.00%	mid-cap
EXOTIC	1000	1.00%	9.90%	long-tail

4 BLOCK-WINDOWED INTEGRAL PRICING

4.1 The cumulative tracker

Pricing each swap by its own δ alone is gameable: an attacker splits one large δ into many small swaps, each read near $\delta = 0$, and pays the base fee repeatedly. To close this, every pool keeps a one-slot window: a `windowStart` block and a signed cumulative of the block's deltas. `BLOCK_WINDOW` is set immutably per chain, so the same wall-clock horizon is covered regardless of block time. The cumulative resets on the first swap of a new window, and each swap is then priced over the interval $(c_0, c_1) = (\text{cumBefore}, \text{cumAfter})$ rather than in isolation. Because the slot is keyed to the *pool*, this holds across every swap on it in the window (regardless of caller, contract, or transaction), so the mechanism cannot be evaded by spreading a trade over many addresses.

4.2 Integral-mode marginal fee

Given the interval (c_0, c_1), the fee is dispatched on how the swap moves the cumulative. A **growth** swap (same sign, larger magnitude, pushing the pool further from neutral) is charged the average of the curve over the path,

$$\text{marginal} \;=\; \frac{1}{|c_1| - |c_0|} \int_{|c_0|}^{|c_1|} \mathrm{fee}(x)\;\mathrm{d}x; \tag{4}$$

an **unwind** (same sign, smaller magnitude, a trade that heals the pool's price) pays only the base fee; and a **flip** (opposite signs) weight-averages the base fee over the unwind half and the integral over the growth half. Pricing by the average of the curve over the interval traversed, rather than by the curve at a single point, is what gives the schedule its two key properties.

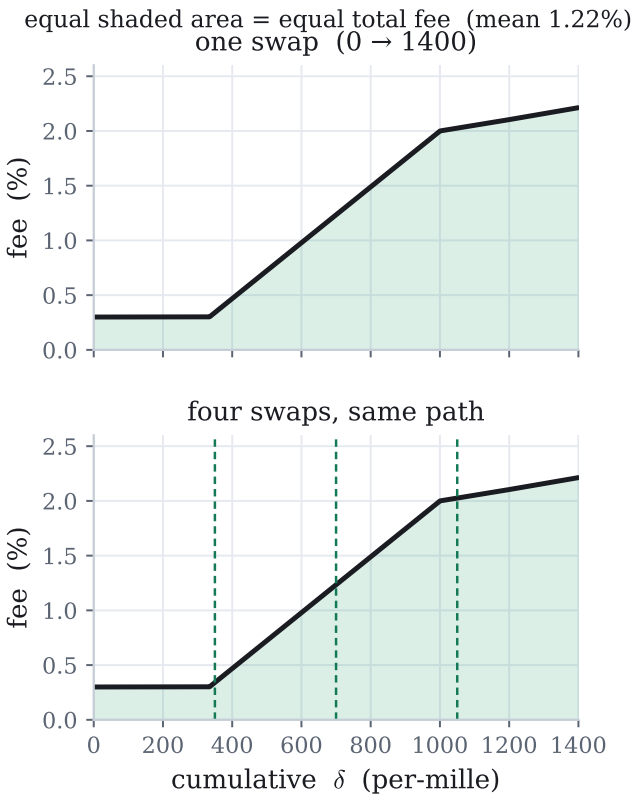

Figure 2. Path-independence. The shaded area (the total fee) is identical whether the cumulative move is taken in one swap or in four sub-swaps over the same interval.

4.3 Path-independence and splitting-resistance

Path-independence. Over a fixed, same-direction move of the cumulative from a to b, the total fee is $\int_a^b \text{fee}$, independent of how the move is sliced. This is immediate from (4): each sub-swap contributes $\int$ over its own sub-interval, and consecutive sub-intervals telescope to $\int_a^b$. Splitting one swap into a same-block bundle therefore changes the total by nothing (Figure 2). As an example, a swap moving a BLUE-CHIP pool from neutral to $\delta = 1200$ pays $\approx 1.06\%$, and any same-block split of that move pays the same $\approx 1.06\%$.

Splitting-resistance. An attacker controls token amounts, not the cumulative directly, and splitting a fixed token rebalance drives the cumulative *deeper*, because each sub-swap is priced against the reserve left by the previous one, so the per-swap deltas sum to more than the single-swap delta. The split therefore traverses a deeper interval and pays the curve over all of it: at least as much as a single swap, and strictly more once it is split. Any partial reversal is penalised further, since the unwind leg pays the base fee while the re-growth leg re-pays the curve. The cheapest way to achieve a given rebalance is thus a single honest swap.

5 ARCHITECTURE AND IMPLEMENTATION

Spry is a small periphery over the canonical v4 `PoolManager`: one hook, one swap-only router, and three stateless libraries (the fee math, the parameter struct, and the virtual-reserve conversion). On each swap the hook reads price and liquidity, computes δ, updates the pool's cumulative window, asks the fee library for the integral-mode marginal fee, and returns it as the LP-fee override. The hook holds no funds and makes no external calls; liquidity is provided separately, through Uniswap's canonical `PositionManager`, with per-position fee accounting [6]. Figure 3 shows the call-flow. The separation is deliberate: the router is the only contract that moves user funds, and it does so only inside v4's unlock; the hook is read-plus-one-write and never custodies value; the fee math is pure. The contract that *can* hold funds, the one that *sets* fees, and the audited core that *settles* are kept distinct.

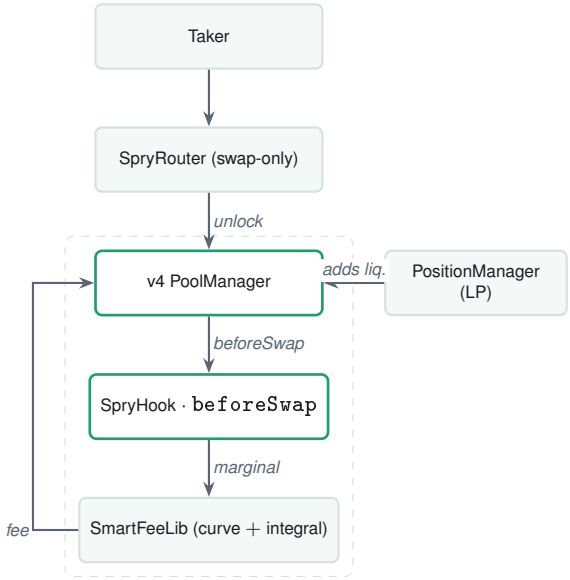

Figure 3. A swap flows taker → router → `PoolManager` → hook → fee library, and the fee returns to the manager as an LP-fee override. The dashed box is the canonical v4 core together with the Spry hook and fee library; liquidity is supplied separately through `PositionManager`.

Implementation notes. v4 derives a hook's permissions from its address, so the hook is deployed at a CREATE2 salt mined until the low bits equal the `BEFORE_SWAP` flag. A Spry pool is initialised with the dynamic-fee flag and one of the five sanctioned `tickSpacing` values; any other is rejected, which binds the pool to exactly one tier. The fee library works on virtual reserves $R_0 = L\,2^{96}/\sqrt{P}_{X96}$ and $R_1 = L\sqrt{P}_{X96}/2^{96}$, computed with 512-bit intermediates to stay exact at extreme prices. The router settles each swap inside a single `PoolManager` unlock callback, with native ETH, multi-hop, and Permit2 support; v4's transient-storage lock [5] guarantees one active unlock, so reentrancy into a second swap during settlement is precluded by the core rather than by bespoke guards. The cumulative is per-pool, so a multi-hop swap through several pools updates each pool's own window independently.

6 RELATED WORK

Constant-product market making, LP returns, and impermanent loss have been studied widely [10, 13–20]. Static-tier pools (Uniswap v1/v2/v3 [1–3]) charge a flat fee per pool and leave the size mismatch unad-

dressed. Auction-managed AMMs such as am-AMM [9] auction the right to set the fee and capture arbitrage, redistributing MEV through the auction rather than a per-swap curve. Oracle- or volatility-driven dynamic fees adjust the fee from an external signal, and LVR-aware designs [8] target the rebalancing loss directly, often via auctions or oracle-referenced pricing. Spry occupies a distinct point: it is endogenous (reading only the pool's own price and liquidity), per-swap and path-priced through the integral-mode curve, and deployed purely as periphery over audited core.

7 SUMMARY

Spry is a non-custodial, permissionless fee mechanism for Uniswap v4 that turns impermanent loss into a priced quantity: each swap pays in proportion to the loss it inflicts, integrated over the block's running price move. Integral-mode pricing makes the schedule path-independent and, because splitting deepens the cumulative, splitting-resistant, so the multicall loophole is closed by construction and the excess fee accrues to LPs through v4's standard channel. The mechanism is a single hook plus a swap-only router that holds no funds, reads no oracle, and rides on the audited v4 core. Natural next steps are an independent audit, a concentrated-liquidity generalisation of the full-range economic model, and empirical calibration of the tier anchors against historical pool flow.

References

[1] H. Adams. 2018. Uniswap v1 Core.

[2] H. Adams, N. Zinsmeister, D. Robinson. 2020. Uniswap v2 Core.

[3] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, D. Robinson. 2021. Uniswap v3 Core.

[4] H. Adams, M. Salem, N. Zinsmeister, et al. 2024. Uniswap v4 Core.

[5] A. Akhunov, M. Salem. 2018. EIP-1153: Transient storage opcodes. Ethereum Improvement Proposals.

[6] JT Riley, et al. 2023. ERC-6909: Minimal Multi-Token Interface. Ethereum Improvement Proposals.

[7] P. R. Berg. PRB-Math: fixed-point math in Solidity.

[8] J. Milionis, C. C. Moallemi, T. Roughgarden, A. L. Zhang. 2022. Automated Market Making and Loss-Versus-Rebalancing. arXiv:2208.06046.

[9] A. Adams, C. C. Moallemi, S. Reynolds, D. Robinson. 2024. Am-AMM: An Auction-Managed Automated Market Maker. arXiv:2403.03367.

[10] G. Angeris, T. Chitra, A. Evans. 2022. When does the tail wag the dog? Curvature and market making. Cryptoeconomic Systems.

[11] A. Aigner, G. Dhaliwal. 2021. Uniswap: Impermanent loss and risk profile of a liquidity provider. arXiv:2106.14404.

[12] A. Park. 2023. The conceptual flaws of decentralized automated market making. Management Science 69(11).

[13] A. Khakhar, X. Chen. 2022. Delta hedging liquidity positions on automated market makers. arXiv:2208.03318.

[14] M. Hafner, H. Dietl. 2024. Impermanent loss conditions: an analysis of decentralized exchange platforms. arXiv:2401.07689.

[15] P. Bergault, L. Bertucci, D. Bouba, O. Guéant. 2023. Automated market makers: mean-variance analysis of LP payoffs and design of pricing functions. Digital Finance.

[16] V. Mohan. 2022. Automated market makers and decentralized exchanges: a DeFi primer. Financial Innovation 8(1).

[17] S. Loesch, N. Hindman, M. B. Richardson, N. Welch. 2021. Impermanent loss in Uniswap v3. arXiv:2111.09192.

[18] D. Miori, M. Cucuringu. 2024. Clustering Uniswap v3 traders from their activity on multiple liquidity pools, via novel graph embeddings. Digital Finance.

[19] E. Bayraktar, A. Cohen, A. Nellis. 2024. DEX specs: a mean field approach to DeFi currency exchanges. arXiv:2404.09090.

[20] C. Alexander, X. Chen, J. Deng, Q. Fu. 2023. Market efficiency improvements from technical developments of decentralized crypto exchanges. SSRN 4495589.

DISCLAIMER

This paper is for general information only and is not investment, financial, legal, tax, or accounting advice, nor an offer to buy or sell any asset. The smart contracts it describes are an unaudited research prototype, are not deployed to any production network, and should not custody material funds prior to an independent audit. Decentralised-finance protocols carry significant and potentially total risk of loss. The fee curves, parameters, and mechanisms here may change and may differ from any deployed implementation; the views are the authors' own, and readers are responsible for their own due diligence and legal compliance.