

LANDAUSWAP RESISTANT AMM â WHITEPAPER

Version: 0.9 (Draft)

Network focus: Solana mainnet-beta

Authors: LandauSwap Core Contributors

Last updated: 2024-XX-XX

0. Abstract

Constant-product AMMs have unlocked permissionless trading but exhibit two structural weaknesses: micro trades pay non-zero slip, while large trades transfer value from LPs to arbitrageurs through leverage-driven rebalancing (LVR). LandauSwap introduces a resistance-based AMM for Solana that bends the price impact curve, providing near-zero slip for retail order flow and steeply escalating fees for whale trades that consume depth. Orders are aggregated into discrete batches, neutralizing sandwich attacks and letting LPs capture the convex resistance premium. The protocol is built around audited SPL vault custody, deterministic on-chain math, and a production architecture that meets aggregator requirements for latency, price previews, and observability.

1. Background & Problem Statement

Automated market makers leveraging constant-product invariants (' $xy = k$ ') have become liquidity backbones across DeFi. Despite their simplicity, two deficiencies persist:

- â¢ **LVR and value leakage:** arbitrageurs profit whenever the pool price lags the true market price, eroding LP returns. Research on Uniswap v2 and v3 shows that up to 50% of swap fees can be siphoned through LVR in volatile markets.
- â¢ **Inefficient micro trades:** even minimal trades experience slippage proportional to the product curve gradient, hurting wallets and payment apps that rely on precise execution.

On Solana, high throughput magnifies MEV susceptibility. The network's short block times allow sandwich bots to front-run trades by inserting transactions before and after a victim swap. Aggregators such as Jupiter and Raydium require pools to deliver consistent price previews, reliable settlement, and real-time data feeds; early-stage AMMs struggle to satisfy these integration thresholds.

LandauSwap targets stable and low-volatility pairs (USDC/USDT, SOL stables) plus long-tail assets that currently lack sustainable LP yield. The design excludes off-chain oracles to avoid centralization, instead relying on batch clearing and convex fees to restore LP edge. The protocol must honor Solana's compute limits and provide fairness to small

traders while maintaining permissionless access.

2. Design Goals & Principles

1. **Capital protection:** LP principal should not be extracted by LVR; resistance fees must accumulate to pool vaults.
2. **Retail friendliness:** trades below a configurable threshold experience slip approaching zero, enabling payments and routing micro-flows.
3. **Convex cost for depth:** large trades pay exponentially higher fees that compensate LPs for temporary imbalance and discourage manipulation.
4. **Permissionless and oracle-free:** all pricing logic resides on-chain without external price feeds; batching ensures coherent clearing prices.
5. **Governable but open:** protocol upgrades and parameter changes go through multi-sig governance while keeping daily operations trustless.
6. **Observability and composability:** every state change emits events; SDKs expose read/write flows so aggregators can integrate seamlessly.

3. Mathematical Model

3.1 Definitions

Consider a pool with reserves '(X, Y)' representing tokens A and B. A trader submits ' x ' of token A intending to receive token B. Define:

- â¢ ' $\phi = x / X$ ' (normalized trade size),
- â¢ ' $r = \phi / 2$ ' (resistance variable),
- â¢ ' $p = Y / X$ ' (instantaneous price before the trade),
- â¢ ' $s(r)$ ' (resistance function, $0 \leq r < 1$).

We assume continuous reserves, discrete batch settlements, and no exogenous price oracle. Batches collect same-direction orders before executing a deterministic clearing function.

3.2 Resistance Curves

The baseline Landau resistance curve is:

[code block omitted]

$$s(r) = r^2 / (1 + r^2)$$

[code block omitted]

This curve satisfies:

- â¢ ' $s(0) = 0$ ' ensuring zero slip at zero trade size,
- â¢ monotonic increase with ' r ',

↳ asymptotic approach to 1, meaning full trade value is captured as fee when the pool is nearly drained,

↳ ' $d^2 s / dr^2 > 0$ ', delivering convex growth.

An alternative exponential family can be toggled by governance:

[code block omitted]

$s_{exp}(r) = 1 - \exp(-r^2)$

[code block omitted]

Hybrid curves blending rational and exponential terms enable tailoring to pool volatility.

All candidate curves must stay within ' $0 \leq s(r) < 1$ ' and be differentiable to maintain predictable slip profiles.

3.3 Execution Price & Fees

For a submitted ' Δx ', the actual output is:

[code block omitted]

$\Delta y_{actual} = (1 - s(r)) * p * \Delta x$

[code block omitted]

while the resistance fee is:

[code block omitted]

$fee = s(r) * p * \Delta x$

[code block omitted]

Fees accumulate in the output token vault. Because batches aggregate flow, ' r ' is computed using the net inflow within the batch, not per individual order. This prevents small orders from being penalized when executed alongside large whales: the entire batch pays the correct blended fee, and pro-rata adjustments ensure fairness.

3.4 Comparison with CPMM

A CPMM delivers slippage roughly proportional to ' $\Delta x / X$ '. In LandauSwap the slip grows quadratically:

[code block omitted]

$slip_resistance \propto (\Delta x / X)^2$ when $\Delta x \ll X$

[code block omitted]

For ' $\Delta x = 0.1\%$ ' of the pool, CPMM slip is ~10 bps, while Landau resistance results in slip < 1 bp. For ' $\Delta x = 20\%$ ', CPMM slip is ~25%, whereas resistance absorbs >60% of the trade as fee, converting depth consumption into LP revenue rather than lost arbitrage opportunity.

3.5 Edge Cases

↳ **Low reserves:** guard rails abort trades when post-trade reserves drop below minimal thresholds. Resistance curves enforce fee saturation, preventing full depletion.

↳ **Asymmetric liquidity:** curves operate on normalized ratios, so imbalanced pools still deliver predictable fees; governance may adjust curve parameters to rebalance incentives.

↳ **Batch exposure:** the protocol sets maximum batch duration (slots) to limit exposure

to external price drift. Orders can specify an expiry slot; expired orders are cancelled and funds returned.

4. Mechanism Design

4.1 Batch Clearing

Orders are accepted continuously but settled at discrete windows. The settler (any keepers or aggregators) calls ‘settle_batch’ once:

- â¢ Net inflow exceeds a volume threshold,
- â¢ Batch duration exceeds ‘min_batch_slots’,
- â¢ Or a manual trigger is provided by governance during low volume.

All trades in a batch clear at the same resistance-adjusted price. Because inflows do not immediately alter reserves, sandwich bots cannot front-run individual orders. Settlers earn a small rebate or share of resistance fees, incentivizing timely execution. Failure to settle allows other actors to step in; the protocol enforces idempotence to avoid double-counting.

4.2 Liquidity Management

LPs deposit SPL tokens into program-owned vaults and receive LP shares minted via the resistance-aware invariant. Withdrawals burn shares and transfer tokens back proportionally, including accrued fees. Fee distribution may be toggled between:

- â¢ **Compounded mode:** fees stay in the vault, increasing the value of each share.
- â¢ **Harvest mode:** governance routes a percentage to protocol treasury or incentives.

LPs can specify min-acceptable output upon deposit/withdrawal, preventing slippage from curve parameter changes mid-transaction. Rebalancing bots and auto-compounding strategies integrate through SDK hooks.

4.3 Integration with Aggregators

Aggregators query a quote service that mirrors on-chain reserves and curve parameters.

Quotes include:

- â¢ expected output,
- â¢ estimated resistance fee,
- â¢ effective price impact,
- â¢ settlement deadline.

Signed RFQ responses can be delivered to aggregators for deterministic routing. The program emits event logs ('OrderPlaced', 'BatchSettled', 'LiquidityChanged') enabling

subgraph/indexers to keep caches up to date. Cross-venue arbitrage is limited by the convex fee curve; nonetheless, integrators can request partial fills or fall back to other pools when Landau resistance deems a trade too large.

5. Implementation Architecture (Solana + Anchor)

5.1 Account Layout

Each pool uses Program Derived Addresses (PDAs) for custody and metadata:

- â¢ ‘Pool’: stores authority, mints, vaults, reserves, accumulated fees, curve type, and governance flags.
- â¢ ‘VaultA’/‘VaultB’: SPL Token accounts owned by the program to hold liquidity.
- â¢ ‘FeeVault’ (optional): collects protocol share of resistance fees.
- â¢ ‘BatchState’: tracks outstanding net deltas, order count, last slot, and curve snapshot.
- â¢ ‘LpMint’: SPL mint representing LP shares; metadata records total supply and fee mode.
- â¢ ‘Config’: global parameters (governance keys, pauses, supported curves).

Account sizes are padded for future fields, aligned to 8-byte boundaries to satisfy rent exemption.

5.2 Instruction Flow

1. ‘initialize_pool’: authority signs, pools PDAs derived via seeds ‘[POOL_SEED, mintA, mintB]’. Vaults and LP mint are created, curve parameters registered.
2. ‘deposit_liquidity’: transfers user funds into vault PDAs, mints LP shares, updates reserves and events.
3. ‘place_order’: verifies escrow token account, locks funds, records batch entry with direction, amount, expiry.
4. ‘settle_batch’: calculates net deltas, runs ‘compute_resistance_trade’, releases output tokens, records fees and batch id.
5. ‘withdraw_liquidity’: burns LP shares, transfers tokens, accounts for outstanding settlement fees.
6. ‘admin_update_config’: multi-sig controlled changes to parameters, emergency pauses, curve toggles.

All instructions include strict signer checks, PDA verification, and optional replay protection (recent slot check).

5.3 Performance & Compute

Solana imposes ~200k compute units per transaction. The resistance math is O(1) and fits well under 10k CUs. Batching ensures few large transactions rather than many small ones.

Parallel pools operate in separate PDAs, so concurrent transactions do not conflict. If settlement fails (e.g., compute overrun), the batch is retried with reduced chunk size.

5.4 Security Surfaces

- â¢ **Access control:** only governance can change config; all other flows remain permissionless.
- â¢ **Vault validation:** every transfer compares expected and actual vault balances to detect flash-loan interference.
- â¢ **Batch manipulation:** orders include expiry slots; stale orders are cancelled automatically to stop griefing.
- â¢ **Upgrade path:** governed by a multi-sig upgrade authority that publishes audits and migration scripts before deployment.
- â¢ **Event indexing:** tamper-evident logs allow watchdogs to reconstruct state and detect anomalies.

6. Evaluation & Testing

The evaluation plan spans simulation, on-chain testing, and security reviews:

- â¢ **Curve simulations:** offline Python/Rust suites replay historical order flow against CPMM and Landau resistance curves, measuring slip, LP yield, and volatility.
- â¢ **Localnet integration tests:** Anchor-based suites cover instruction invariants, batch settlement, fee accounting, and failure cases.
- â¢ **Fuzzing/property tests:** QuickCheck-style fuzzers explore boundary conditions (minimal reserves, extreme trade sizes).
- â¢ **Load testing:** devnet deployment with bots generating thousands of orders per minute to profile compute usage and latency.
- â¢ **Audit pipeline:** third-party audits focus on program safety, math correctness, and governance mechanisms; bug bounty launches alongside guarded mainnet.

Key metrics include LP APY vs. CPMM baseline, fee capture distribution, average settlement latency, failure rate per batch, and aggregator routing success.

7. Roadmap & Deployment Plan

1. **MVP Hardening (now):** finalize SPL custody, LP share accounting, rational curve, SDK quoting prototype.
2. **Testnet Beta (T+4 weeks):** deploy to Solana devnet, onboard early LPs, integrate with at least one aggregator sandbox, run audit.
3. **Guarded Mainnet Launch (T+8 weeks):** cap TVL, enable whitelisted settlers, introduce protocol fee split.

4. **Curve Extensions (T+12 weeks):** add exponential/hybrid curves, adaptive fee bands governed by DAO.
5. **Ecosystem Expansion (T+20 weeks):** release analytics dashboard, auto-rebalancing bots, liquidity mining programs.
6. **DAO Transition (T+28 weeks):** hand governance to community-controlled DAO, expand compliance tooling for institutional partners.

Each milestone triggers documentation updates, SDK releases, and community announcements. Emergency procedures and rollback scripts are rehearsed before mainnet launch.

8. Tokenomics (If Applicable)

LandauSwap can operate without a native token; however, a protocol token unlocks additional alignment:

- â¢ **Utility:** staking for governance, collateral for settler rewards, fee discounts for routing partners.
- â¢ **Distribution:** bootstrap LP mining, grant shares to core contributors and ecosystem partners with multi-year vesting.
- â¢ **Fee policy:** resistance fees split into ‘LP share’, ‘protocol treasury’, and optional ‘insurance fund’. The treasury may buy back and burn tokens or fund further development.
- â¢ **Compliance:** jurisdiction-specific requirements (KYC/AML) are evaluated before enabling token incentives in restricted regions. Governance maintains a modular denylist contract for regulated integrations.

Any token launch is conditioned on legal review and DAO ratification.

9. Risk Analysis & Mitigations

- â¢ **Economic risks:** thin liquidity can amplify resistance fees and deter traders. Mitigation includes targeted incentives, adaptive curves for low-liquidity pools, and aggregator coordination to route partial orders.
- â¢ **Market manipulation:** whales might oscillate between pools to game fees. Batching and convex costs reduce profitability; governance can adjust parameters rapidly via multi-sig.
- â¢ **Technical failures:** Solana congestion or program bugs may lock batches. Watchdog services monitor backlog; pause mechanisms allow withdrawals while trading is frozen.
- â¢ **Keeper concentration:** if few settlers exist, batches may stall. Incentive programs and open SDKs encourage multiple relayers; fallback settlement by governance remains available.
- â¢ **Regulatory shifts:** monitoring frameworks and compliance-ready hooks position the protocol to react quickly to new obligations.

10. References & Related Work

- â¢ Angeris et al., "Improved Price Oracles: Constant Function Market Makers" (Uniswap research).
- â¢ Milionis et al., "Loss Versus Rebalancing: Measuring LVR in AMMs."
- â¢ Cow Protocol, "Batch Auctions and MEV Mitigation."
- â¢ Curve Finance, "Dynamic Fees on Stableswap."
- â¢ Solana Labs, "Compute Budget and Transaction Processing."
- â¢ Algebra Protocol, "Hybrid Liquidity Pools with Adaptive Curves."

Further references will be appended as formal citations once audit reports and simulation whitepapers are published.

Appendices

- â¢ **Appendix A:** Detailed derivation of resistance functions and proofs of convexity (in preparation).
- â¢ **Appendix B:** Simulation methodology, parameter tables, and sample datasets (forthcoming).
- â¢ **Appendix C:** Glossary covering AMM terminology, MEV, LVR, PDA, and batching.
- â¢ **Appendix D:** Change log tracking revisions to this whitepaper.

Change Log

- â¢ **v0.9:** First complete draft aligning architecture blueprint, README positioning, and implementation plan.

Feedback from partners, auditors, and community reviewers is welcome. Pull requests or issues in the repository should tag 'whitepaper' for visibility.