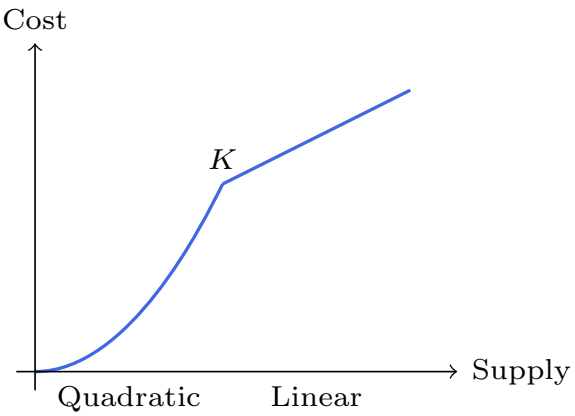
zCurve: Customizable Bonding Curves for AMM Seeding

1. Overview

zCurve is a gas-optimized bonding curve singleton designed to bootstrap zAMM liquidity pools. It implements a *quadratic-then-linear* pricing model that balances early price discovery with sustainable growth, automatically finalizing into full-range AMM positions.



2. Pricing Mechanism

For n tokens (in 10^{12} base units), cost is:

$$C(n) = \begin{cases} \frac{m(m-1)(2m-1)}{6d} \text{ ETH} & m \le K \\ \frac{K(K-1)(2K-1)}{6d} + \frac{K^2(m-K)}{6d} & m > K \end{cases}$$

where $m = \lfloor n/10^{12} \rfloor$, K = quadratic cap, d = divisor.

- **Quadratic**: Progressive discovery, $p_m = m^2/6d$
- **Linear tail**: Constant price $p_K = K^2/6d$

3. Key Features

3.1 Launch Parameters

- **Supply tranches**: Creator, sale, LP ($\le 2^{96}$)
- **Price shaping**: divisor, quadCap
- **Funding target**: ethTarget (wei)
- **Auto-finalize**: On target/deadline/sold out
- **LP handling**: Packed unlock timestamp

3.2 Gas Optimizations

- Binary search for ETH→coins
- Transient reentrancy guard (EIP-1153)
- Granularity quantization (10^{12})

4. Economic Model

4.1 Marginal Price Formula

The marginal price at tick m is:

$$p_m = \begin{cases} \frac{m^2}{6d} \text{ ETH/tick} & m \le K \\ \frac{K^2}{6d} \text{ ETH/tick} & m > K \end{cases}$$

4.2 LP Scaling Algorithm

At finalization, LP tokens are scaled to match spot:

$$\text{LP}_{\text{final}} = \min\left(\text{LP}_{\text{supply}}, \frac{\text{ETH}_{\text{escrow}}}{p_{\text{spot}}} \cdot 10^{12}\right)$$

5. AMM Integration

Upon finalization, zCurve:

1. Scales LP to match spot price
2. Deposits ETH + coins to zAMM
3. Handles LP token distribution

6. Technical Details

6.1 Storage Layout

Slot 1	creator (160) + saleCap (96)
Slot 2	lpSupply (96) + netSold (96) + deadline (64)
Slot 3	divisor (256)
Slot 4	ethEscrow (128) + ethTarget (128)
Slot 5	feeOrHook (256)
Slot 6	quadCap (96) + unused (64) + lpUnlock (96)

Note: `feeOrHook` can encode custom pool logic (e.g., creator taxes, buy cooldowns) via hook contracts.

6.2 Binary Search Implementation

```
while (lo < hi) {
  mid = (lo + hi + 1) >> 1;
  if (_cost(mid) <= value) lo = mid;
  else hi = mid - 1;
}
coinsOut = _quantizeDown(lo);
```

7. Security Considerations

- **Reentrancy**: Transient storage guard
- **Overflow**: Bounded to $2^{96} - 1$ tokens
- **Griefing**: Minimum sale size enforced
- **Timing**: Creator unlock > deadline

8. Future Extensions

- **Custom curves**: Pack hook address into quadCap field for arbitrary pricing via `_cost()` delegation
- **Multi-phase**: Support n-segment piecewise functions
- **Cross-chain**: Deterministic CREATE2 addresses
- **Vesting curves**: Time-weighted price unlocks

9. Design Philosophy

zCurve embraces simplicity over complexity:

- **Predictable math**: Quadratic → linear is intuitive
- **No hidden mechanics**: All parameters visible on-chain
- **Atomic finalization**: One transaction to seed AMM
- **Minimal dependencies**: Only interfaces with zAMM

By avoiding complex curves or multi-phase mechanisms, zCurve remains auditable and gas-efficient while providing effective price discovery.

10. Use Cases

- Fair token launches with price discovery
- Controlled liquidity bootstrapping
- Community-driven project funding
- Automated market making at discovered prices

Precise Discovery. Optimal Seeding. Gas Efficient.

0x00000000007732aBAd9e86BDd0C3A270197EF2e1