PeggedSwap (2-token StableSwap) Additivity of Swaps: (1) Fee-free strict additivity, (2) Superadditivity of the original (Curve-style) output-fee rule, (3) A strictly-additive reinvested-fee construction on top of PeggedSwap

Abstract

We study two-token PeggedSwap mechanisms built from the StableSwap invariant introduced by Egorov [2019]. We first prove that the fee-free StableSwap swap (defined as motion along a constant-D invariant curve) is *strictly additive*: swapping Δ is exactly the same as swapping a then b with $\Delta = a + b$. We then formalize the *original PeggedSwap fee rule used in practice* (Curve-style): compute the no-fee output on the invariant curve, charge a fee on the output, and keep that fee inside the pool. We show that this rule is generally *superadditive* for the trader: splitting a trade into two chunks yields strictly larger total output than doing the same total input in one shot. Finally, we present a general functional equation for strict additivity—drawing on classical results from Aczél [1966]—and construct a strictly-additive fee/reinvest mechanism on top of StableSwap by enforcing a semigroup law on the liquidity scalar D.

Contents

1 Introduction and Related Work

1.1 Background on Constant Function Market Makers

Automated market makers (AMMs) have become fundamental infrastructure in decentralized finance. The mathematical foundations of constant function market makers (CFMMs) were formalized by Angeris and Chitra [2020], who established that CFMMs can be characterized by trading functions $\varphi : \mathbb{R}^n \to \mathbb{R}$ where valid trades satisfy $\varphi(R + \Delta) = \varphi(R)$ for reserves R and trade Δ. The geometric properties of CFMMs were further developed by Angeris et al. [2023], who proved that every CFMM has a unique canonical trading function that is nondecreasing, concave, and homogeneous.

The StableSwap invariant was introduced by Egorov [2019] to provide efficient exchange for stablecoins by interpolating between constant-sum and constant-product behavior. This was later extended to volatile assets via the CryptoSwap mechanism [Egorov, 2021]. Related AMM designs include Balancer's weighted geometric mean invariant [Martinelli and Mushegian, 2019] and Bancor's bonding curves [Hertzog et al., 2018].

1.2 Axiomatic Approaches to AMM Design

Recent work has developed axiomatic foundations for AMMs. Schlegel et al. [2023] introduced the *Liquidity Additivity* axiom for multi-asset settings and characterized constant product market makers via independence and scale invariance. Bichuch and Feinstein [2022] proposed axioms including conditions under which AMMs are indifferent to transaction splitting—a key additivity property. Frongillo et al. [2024] established equivalence between CFMMs with concave potential functions and cost-function prediction markets, with *StrongPathIndependence* as a central axiom.

1.3 Fee Mechanisms and LP Economics

Understanding how fees affect swap outcomes is essential for analyzing additivity. Milionis et al. [2022] introduced Loss-Versus-Rebalancing (LVR) as the fundamental adverse selection cost for liquidity providers, proportional to $\sigma^2 \cdot V''(P)$ (volatility squared times gamma). Optimal fee design has been studied by Evans et al. [2021] for geometric mean market makers and more recently by dynamic fee mechanisms [Baggiani et al., 2025].

1.4 Routing and Composability

The question of how swap outcomes depend on execution paths is directly related to routing optimization. Angeris et al. [2022] formulated routing across CFMM networks as convex optimization, while Angeris et al. [2021] extended this to general multi-asset trades. The composability of DeFi protocols and its relationship to path-independence was formalized by Bartoletti et al. [2024].

1.5 Contributions of This Paper

We make three main contributions:

(1) We prove that fee-free StableSwap is *strictly additive* (Definition 1): the trader output from a single swap equals the total output from any split of that swap.

(2) We prove that the original Curve-style output-fee rule is *superadditive*: splitting yields strictly *more* output than one-shot execution.

(3) We derive the necessary and sufficient condition on fee/reinvest rules for strict additivity—a semigroup law on the liquidity scalar D—connecting to classical functional equation theory [Aczél, 1966].

2 Setup, Additivity Definitions, and the Master Functional Equation

2.1 State and Direction

We consider two reserves

$$(x, y) \in \mathbb{R}^2_{>0},$$

where x is token X reserve and y is token Y reserve. We study swaps in direction $X \to Y$.

2.2 Update Map

A swap rule is a family of maps

$$F_\Delta : \mathbb{R}^2_{>0} \to \mathbb{R}^2_{>0}, \qquad F_\Delta(x, y) = \big(x + \Delta,\ Y(x, y; \Delta)\big),$$

where $\Delta > 0$ is the trader's input amount in token X, and $Y(x, y; \Delta)$ is the post-swap Y-reserve.

2.3 Trader Output

We define trader output as

$$\mathrm{Out}(\Delta; x, y) := y - Y(x, y; \Delta).$$

Larger Out is better for the trader.

2.4 Additivity Types: Strict, Sub-, and Super-additivity

Following standard conventions in economics and game theory [Sharkey, 1982, Shapley, 1953], we define three types of additivity for trader output:

Definition 1 (Additivity Types for Trader Output)**.** Let $\mathrm{Out}_{\mathrm{split}}(a, b; x, y)$ denote the total trader output from executing swap a followed by swap b:

$$\mathrm{Out}_{\mathrm{split}}(a, b; x, y) := \mathrm{Out}(a; x, y) + \mathrm{Out}\big(b; F_a(x, y)\big).$$

The swap mechanism is:

(a) **Strictly additive** if for all $a, b \geq 0$ and all states (x, y):

$$\mathrm{Out}(a + b; x, y) \ = \ \mathrm{Out}_{\mathrm{split}}(a, b; x, y). \tag{1}$$

(One-shot output *equals* split output.)

(b) **Subadditive** if for all $a, b > 0$ and all states (x, y):

$$\mathrm{Out}(a + b; x, y) \ > \ \mathrm{Out}_{\mathrm{split}}(a, b; x, y). \tag{2}$$

(One-shot output is *strictly greater than* split output; splitting is worse for the trader.)

(c) **Superadditive** if for all $a, b > 0$ and all states (x, y):

$$\mathrm{Out}(a + b; x, y) \ < \ \mathrm{Out}_{\mathrm{split}}(a, b; x, y). \tag{3}$$

(One-shot output is *strictly less than* split output; splitting is better for the trader.)

Remark 1 (Mnemonic)**.** The naming convention follows the trader's perspective on the *split* operation:

Type	**Inequality**	**Interpretation**
Strictly additive	$\mathrm{Out}(a + b) = \mathrm{Out}_{\mathrm{split}}$	Path-independent
Subadditive	$\mathrm{Out}(a + b) > \mathrm{Out}_{\mathrm{split}}$	One-shot better
Superadditive	$\mathrm{Out}(a + b) < \mathrm{Out}_{\mathrm{split}}$	Split better

2.5 Semigroup Property and the Master Functional Equation

Definition 2 (Semigroup Property)**.** The swap family $\{F_\Delta\}_{\Delta\geq 0}$ satisfies the *semigroup property* if

$$F_{a+b} \;=\; F_b \circ F_a \qquad \forall\, a, b \geq 0, \tag{4}$$

with $F_0 = \mathrm{Id}$.

The connection between strict additivity (Definition 1(a)) and the semigroup property is fundamental:

Proposition 1 (Equivalence of Strict Additivity and Semigroup Property)**.** *The following are equivalent:*

(i) The swap family $\{F_\Delta\}$ is strictly additive.

(ii) The swap family $\{F_\Delta\}$ satisfies the semigroup property (4)*.*

(iii) The Y-component satisfies the ***master functional equation****:*

$$Y(x, y; a+b) \;=\; Y\Big(x + a,\; Y(x, y; a)\;;\; b\Big) \qquad \forall\, x, y > 0,\; a, b \geq 0. \tag{5}$$

Proof. (i)⇔(ii): By definition, $\mathrm{Out}(a+b) = \mathrm{Out}_{\mathrm{split}}(a, b)$ holds for all states iff $F_{a+b} = F_b \circ F_a$.

(ii)⇔(iii): The x-component of $F_{a+b}(x, y)$ is $x + a + b$, which equals the x-component of $F_b(F_a(x, y)) = F_b(x + a, Y(x, y; a))$, namely $(x + a) + b$. Thus (4) reduces to equality of Y-components, which is (5). □

Remark 2 (Connection to Cauchy's Functional Equation)**.** Equation (5) is a two-parameter generalization of Cauchy's functional equation $f(a + b) = f(a) + f(b)$, which Aczél [1966] showed has only linear solutions $f(x) = cx$ under mild regularity conditions (continuity, monotonicity, or measurability). The structure of (5) will similarly constrain the form of strictly-additive swap mechanisms.

3 Two-token StableSwap Invariant and Explicit GetY

3.1 Invariant Equation

Fix amplification parameter $A > 0$ (pool parameter, constant during swaps). The two-token StableSwap invariant, introduced by Egorov [2019], is defined via a scalar $D > 0$ satisfying:

$$4A(x + y) + D \;=\; 4AD + \frac{D^3}{4xy}. \tag{6}$$

Define

$$I(x, y, D) := 4A(x + y) + D - 4AD - \frac{D^3}{4xy}. \tag{7}$$

For a given state (x, y), the pool's D is determined by solving

$$I(x, y, D) = 0, \qquad D > 0. \tag{8}$$

Remark 3 (Homogeneity)**.** If $(x, y) \mapsto (tx, ty)$ for $t > 0$, then the solution scales as $D \mapsto tD$. Thus D is the natural "pool size" variable for pegged AMMs, as noted by Angeris et al. [2023] in their geometric characterization of CFMMs.

3.2 Explicit Solution for D (Cubic Formula)

The StableSwap invariant (6) can be rearranged into a depressed cubic in D:

$$D^3 - 4xy(4A-1)D - 16Axy(x+y) = 0. \tag{9}$$

Proposition 2 (Closed-Form Solution for D)**.** *For reserves (x, y) with $x, y > 0$ and amplification $A > 0$, the unique positive solution to* (9) *is given by Cardano's formula. Define:*

$$p := -4xy(4A-1), \tag{10}$$

$$q := -16Axy(x+y), \tag{11}$$

$$\Delta := \frac{q^2}{4} + \frac{p^3}{27}. \tag{12}$$

Then:

$$D = \sqrt[3]{-\frac{q}{2} + \sqrt{\Delta}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\Delta}}. \tag{13}$$

Remark 4 (Balanced Pool Simplification)**.** At balance where $x = y$, the solution simplifies dramatically:

$$D = 2x \quad (\text{when } x = y). \tag{14}$$

This can be verified by substitution into (6).

Remark 5 (Numerical Implementation)**.** In practice, D is typically computed iteratively using Newton-Raphson rather than the cubic formula, as iteration converges quickly and avoids numerical issues with cube roots.

Remark 6 (Alternative PeggedSwap Formulation)**.** The implementation in this codebase uses a related but distinct *square-root linear* invariant:

$$\sqrt{u} + \sqrt{v} + A(u+v) = C, \tag{15}$$

where $u = x/X_0$ and $v = y/Y_0$ are normalized reserves, X_0, Y_0 are reference (initial) reserves, A is the linear width parameter, and C is the invariant constant. This formula:

- Admits a closed-form quadratic solution for v given u (see Proposition 4),
- Has D implicitly encoded in (X_0, Y_0) via the scaling: if the pool grows by factor t, then $X_0 \mapsto tX_0$, $Y_0 \mapsto tY_0$,
- Reduces to the StableSwap behavior for pegged assets while enabling analytical solutions.

At balance ($u = v = 1$), the invariant is $C_{\text{bal}} = 2 + 2A$.

3.3 Explicit Solution $y = \text{GetY}(x; D)$ for the Two-token Case

Proposition 3 (Derivation of a Quadratic in y)**.** *Fix $A > 0$ and $D > 0$. For any $x' > 0$, the equation $I(x', y', D) = 0$ is equivalent to*

$$16Ax'\,(y')^2 \;+\; b(x', D)\,y' \;-\; D^3 \;=\; 0, \tag{16}$$

where

$$b(x', D) := 16A(x')^2 + 4Dx' - 16ADx'. \tag{17}$$

Proof. Start from $I(x', y', D) = 0$:

$$4A(x' + y') + D - 4AD - \frac{D^3}{4x'y'} = 0.$$

Move the fraction:

$$4A(x' + y') + D - 4AD = \frac{D^3}{4x'y'}.$$

Multiply both sides by $4x'y'$:

$$4x'y'\big(4A(x' + y') + D - 4AD\big) = D^3.$$

Distribute:

$$16Ax'y'(x' + y') + 4Dx'y' - 16ADx'y' = D^3.$$

Expand $16Ax'y'(x' + y')$:

$$16Ax'y'x' + 16Ax'y'y' = 16A(x')^2y' + 16Ax'(y')^2.$$

So:

$$16Ax'(y')^2 + \big(16A(x')^2 + 4Dx' - 16ADx'\big)y' = D^3.$$

Bring D^3 to the left and identify $b(x', D)$ to obtain (16). □

Proposition 4 (Closed Form GetY)**.** *Fix $A > 0$ and $D > 0$. The physically relevant root of* (16) *is*

$$\mathrm{GetY}(x'; D) \;=\; \frac{-b(x', D) + \sqrt{b(x', D)^2 + 64Ax'D^3}}{32Ax'}. \tag{18}$$

Proof. Quadratic formula for $a(y')^2 + by' + c = 0$ with

$$a = 16Ax', \quad b = b(x', D), \quad c = -D^3$$

gives

$$y' = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{b^2 + 64Ax'D^3}}{32Ax'}.$$

The branch with $-b + \sqrt{\cdot}$ yields the positive, continuous solution for the stable AMM curve; this is (18). □

4 (1) Fee-free PeggedSwap is Strictly Additive

4.1 Fee-free Update Map

Definition 3 (Fee-free StableSwap Map S_Δ)**.** Given state (x, y), compute D_0 from $I(x, y, D_0) = 0$. For input $\Delta \geq 0$ define

$$S_\Delta(x, y) := \big(x + \Delta,\ y_\Delta\big), \qquad y_\Delta := \mathrm{GetY}(x + \Delta;\ D_0). \tag{19}$$

Trader output:

$$\mathrm{Out}_{\mathrm{nofee}}(\Delta; x, y) := y - y_\Delta.$$

4.2 Strict Additivity Theorem

Theorem 1 (Strict Additivity of Fee-free PeggedSwap)**.** *The fee-free StableSwap mechanism (Definition 3) is strictly additive:*

$$\mathrm{Out}_{\mathrm{nofee}}(a+b;x,y)=\mathrm{Out}_{\mathrm{nofee}}(a;x,y)+\mathrm{Out}_{\mathrm{nofee}}\big(b;S_a(x,y)\big)$$

for all $a,b\geq 0$ and all states (x,y). Equivalently, $S_{a+b}=S_b\circ S_a$.

Proof with All Intermediate Steps. Fix a starting state (x,y) and let D_0 be the unique solution of $I(x,y,D_0)=0$.

Step 1 (One-shot). By (19),

$$S_{a+b}(x,y)=\Big(x+a+b,\ \mathrm{GetY}(x+a+b;\,D_0)\Big).$$

Step 2 (Two-step). First apply S_a:

$$S_a(x,y)=\Big(x+a,\ \mathrm{GetY}(x+a;\,D_0)\Big).$$

Now apply S_b to the result. In the fee-free model, the swap is defined as motion on the *same* constant-D_0 curve, so S_b uses the same D_0:

$$S_b(S_a(x,y))=\Big((x+a)+b,\ \mathrm{GetY}((x+a)+b;\,D_0)\Big)=\Big(x+a+b,\ \mathrm{GetY}(x+a+b;\,D_0)\Big).$$

Step 3 (Equality). The expressions in Step 1 and Step 2 match exactly. Hence $S_{a+b}=S_b\circ S_a$, establishing the semigroup property. By Proposition 1, this implies strict additivity. □

Remark 7 (Geometric Interpretation)**.** The strict additivity of fee-free StableSwap follows from the fact that all swaps trace the same constant-D level curve. This is an instance of the path-independence property characterized by Angeris et al. [2023] for CFMMs with convex trading sets.

5 (2) Original PeggedSwap Fee (Curve-style) is Superadditive

5.1 Definition of the Original Output-Fee Rule (Fee Kept in the Pool)

The *original StableSwap fee logic* used in practice for pegged pools can be described as:

(a) compute the *fee-free* output Δy_{nf} on the constant-D curve,

(b) charge fee f on the output amount,

(c) pay only $(1-f)\Delta y_{\mathrm{nf}}$ to the trader,

(d) keep $f\Delta y_{\mathrm{nf}}$ inside the pool as extra Y.

We formalize this exactly.

Definition 4 (Curve-style Output-Fee Map $F_{\Delta}^{\mathrm{out}}$)**.** Fix fee rate $f\in(0,1)$. Given state (x,y):

(1) Compute D_0 from $I(x,y,D_0)=0$.

(2) Compute fee-free post-swap reserve:

$$y_{\text{nf}} := \text{GetY}(x + \Delta;\, D_0).$$

(3) Fee-free output amount:

$$\Delta y_{\text{nf}} := y - y_{\text{nf}}.$$

(4) Trader receives (fee charged on output):

$$\text{Out}_{\text{outfee}}(\Delta; x, y) := (1 - f)\, \Delta y_{\text{nf}}.$$

(5) Pool keeps the fee in Y, so final Y reserve is

$$y' := y - \text{Out}_{\text{outfee}}(\Delta; x, y) = y - (1 - f)(y - y_{\text{nf}}) = y_{\text{nf}} + f(y - y_{\text{nf}}).$$

Define

$$F_{\Delta}^{\text{out}}(x, y) := \big(x + \Delta,\ y'\big). \tag{20}$$

Remark 8 (Convenient Closed Form for y')**.** From the last line above,

$$y' = (1 - f)\, y_{\text{nf}} + f\, y. \tag{21}$$

So y' is an affine combination of the no-fee y_{nf} and the starting y. In particular, since $y_{\text{nf}} < y$ for $\Delta > 0$, we have

$$y_{\text{nf}} < y' < y.$$

5.2 Why F_{Δ}^{out} is Not Strictly Additive (Path Dependence)

Because y' in (21) is *not* equal to the fee-free y_{nf}, the post-swap state leaves the original constant-D_0 curve. The next chunk recomputes a new D, so the overall map depends on how a trade is split. This path dependence, as noted by Bichuch and Feinstein [2022], is a key feature distinguishing practical AMM implementations from idealized models.

5.3 A Clean Decomposition Identity for Comparing One-shot vs Split

Let $\Delta = a + b$. Start from (x_0, y_0) with D_0.

Define the fee-free intermediate point on the *same* constant-D_0 curve:

$$\bar{y} := \text{GetY}(x_0 + a;\, D_0). \tag{22}$$

And the fee-free final point for one-shot amount $a + b$:

$$y_{\text{nf}}^{a+b} := \text{GetY}(x_0 + a + b;\, D_0). \tag{23}$$

Then the fee-free output telescopes *exactly*:

$$(y_0 - y_{\text{nf}}^{a+b}) \;=\; (y_0 - \bar{y}) + (\bar{y} - y_{\text{nf}}^{a+b}). \tag{24}$$

This identity is the key to a short superadditivity proof.

5.4 Monotonicity Assumption Needed for a Rigorous Sign Theorem

To prove a general superadditivity inequality, we need a single monotonicity fact:

Definition 5 (Monotone-in-y Property for No-fee Output)**.** Fix $b > 0$ and consider the fee-free StableSwap output for input b:

$$\mathrm{Out}_{\mathrm{nofee}}(b; x, y) := y - \mathrm{GetY}(x + b;\, D(x, y)),$$

where $D(x, y)$ is defined by $I(x, y, D) = 0$. We say the mechanism is *monotone in* y if for fixed x and b the function

$$y \mapsto \mathrm{Out}_{\mathrm{nofee}}(b; x, y)$$

is nondecreasing.

Remark 9 (Why This Monotonicity is Natural)**.** If you hold x fixed and increase y, the pool has more Y liquidity (and higher D), so it should not pay out less Y for the same X input. This monotonicity is satisfied in the pegged region for StableSwap and can be verified numerically (see §5.6). Similar monotonicity properties are implicit in the CFMM characterization of Angeris and Chitra [2020].

5.5 Superadditivity Theorem (Split is Better) for the Original Output-Fee Rule

Theorem 2 (Superadditivity of Curve-style Output-Fee)**.** *Assume the monotone-in-y property from Definition 5 holds for the states visited by the trade. Then for any $a, b > 0$ and any starting state (x_0, y_0), the original output-fee rule (Definition 4) is* superadditive for the trader*:*

$$\mathrm{Out}_{\mathrm{outfee}}(a + b;\, x_0, y_0) \;<\; \mathrm{Out}_{\mathrm{outfee}}(a;\, x_0, y_0) \;+\; \mathrm{Out}_{\mathrm{outfee}}\big(b;\, F_a^{\mathrm{out}}(x_0, y_0)\big). \tag{25}$$

Equivalently: splitting into a then b yields strictly more total output than one-shot.

Proof with Explicit Intermediate Steps. Fix $a, b > 0$ and (x_0, y_0).

Step 1 (One-shot output). Compute D_0 from $I(x_0, y_0, D_0) = 0$. Fee-free final reserve on the D_0 curve is y_{nf}^{a+b} as in (23). So

$$\Delta y_{\mathrm{nf}}^{a+b} = y_0 - y_{\mathrm{nf}}^{a+b}.$$

By Definition 4, the trader output in one shot is

$$\mathrm{Out}_{\mathrm{outfee}}(a + b;\, x_0, y_0) = (1 - f)\,(y_0 - y_{\mathrm{nf}}^{a+b}). \tag{26}$$

Step 2 (Split: first chunk a). Fee-free intermediate Y reserve (on the same D_0 curve) is $\bar{y}$ from (22). Fee-free output for chunk a is $y_0 - \bar{y}$. Trader receives

$$\mathrm{Out}_1 := \mathrm{Out}_{\mathrm{outfee}}(a;\, x_0, y_0) = (1 - f)\,(y_0 - \bar{y}). \tag{27}$$

Pool keeps fee in Y, so after the first chunk, the actual Y reserve becomes

$$y_1 = \bar{y} + f(y_0 - \bar{y}) = (1 - f)\bar{y} + f y_0. \tag{28}$$

Note from $f \in (0, 1)$ and $\bar{y} < y_0$ that

$$y_1 > \bar{y}. \tag{29}$$

Step 3 (One-shot decomposes into two fee-free segments). From the telescoping identity (24),

$$y_0 - y_{\text{nf}}^{a+b} = (y_0 - \bar{y}) + (\bar{y} - y_{\text{nf}}^{a+b}).$$

Multiply by $(1 - f)$:

$$(1 - f)(y_0 - y_{\text{nf}}^{a+b}) = (1 - f)(y_0 - \bar{y}) + (1 - f)(\bar{y} - y_{\text{nf}}^{a+b}). \tag{30}$$

The first term is exactly Out_1 from (27). Thus,

$$\text{Out}_{\text{outfee}}(a + b;\, x_0, y_0) = \text{Out}_1 + (1 - f)(\bar{y} - y_{\text{nf}}^{a+b}). \tag{31}$$

Step 4 (Interpret the second term as a fee-free output from the virtual state). Consider the *virtual* state $(x_0 + a, \bar{y})$ (which lies on the constant-D_0 curve by definition of $\bar{y}$). If we apply a fee-free swap of size b starting from $(x_0 + a, \bar{y})$, the fee-free post-swap Y reserve is exactly $y_{\text{nf}}^{a+b} = \text{GetY}(x_0 + a + b; D_0)$. Thus, the fee-free output for that virtual second leg is

$$\text{Out}_{\text{nofee}}\big(b;\, x_0 + a, \bar{y}\big) = \bar{y} - y_{\text{nf}}^{a+b}.$$

Therefore, the second term in (31) is

$$(1 - f)(\bar{y} - y_{\text{nf}}^{a+b}) = (1 - f)\,\text{Out}_{\text{nofee}}(b;\, x_0 + a, \bar{y}). \tag{32}$$

Step 5 (Compare virtual second leg vs actual second leg using monotonicity in y). The *actual* state before executing the second chunk in the split path is $(x_0 + a, y_1)$ where $y_1 > \bar{y}$ by (29). By Definition 5 (monotone in y for fee-free output), we have

$$\text{Out}_{\text{nofee}}(b;\, x_0 + a, y_1) \;\geq\; \text{Out}_{\text{nofee}}(b;\, x_0 + a, \bar{y}). \tag{33}$$

Step 6 (Actual second-leg output with output-fee). By Definition 4, the trader receives on the second chunk:

$$\text{Out}_2 := \text{Out}_{\text{outfee}}\big(b;\, x_0 + a, y_1\big) = (1 - f)\,\text{Out}_{\text{nofee}}(b;\, x_0 + a, y_1). \tag{34}$$

Combine (33) and (34):

$$\text{Out}_2 \;\geq\; (1 - f)\,\text{Out}_{\text{nofee}}(b;\, x_0 + a, \bar{y}). \tag{35}$$

Step 7 (Finish). From (31) and (32),

$$\text{Out}_{\text{outfee}}(a + b;\, x_0, y_0) = \text{Out}_1 + (1 - f)\,\text{Out}_{\text{nofee}}(b;\, x_0 + a, \bar{y}).$$

Using (35),

$$\text{Out}_{\text{outfee}}(a + b;\, x_0, y_0) < \text{Out}_1 + \text{Out}_2.$$

This is exactly (25), establishing superadditivity. □

5.6 Numerical Verification of Superadditivity

We now *check with numbers* that the original output-fee rule is superadditive.

Example 1 (Superadditivity at 1000/1000, $A = 100$, $f = 0.3\%$)**.** Take

$$(x_0, y_0) = (1000, 1000), \qquad A = 100, \qquad f = 0.003.$$

At balance, the invariant gives $D_0 = x_0 + y_0 = 2000$.

We compute GetY via (18).

One-shot $\Delta = 200$. Fee-free:

$$y_{\text{nf}} = \text{GetY}(1200; 2000) \approx 800.2069861160427, \quad \Delta y_{\text{nf}} = 1000 - y_{\text{nf}} \approx 199.79301388395731.$$

Trader output with output-fee:

$$\text{Out}_{\text{outfee}}(200) = (1 - f)\Delta y_{\text{nf}} \approx 0.997 \cdot 199.79301388395731 \approx 199.19363484230544.$$

Split $100 + 100$.

First 100:

$$y_{\text{nf},1} = \text{GetY}(1100; 2000) \approx 900.0502232299245, \quad \Delta y_{\text{nf},1} \approx 99.94977677007546,$$

$$\text{Out}_1 = 0.997 \cdot 99.94977677007546 \approx 99.64992743976524,$$

$$y_1 = y_{\text{nf},1} + f\Delta y_{\text{nf},1} \approx 900.0502232299245 + 0.003 \cdot 99.94977677007546 \approx 900.3500725602348.$$

Second 100 (recompute D_1 from $(1100, y_1)$):

$$D_1 \approx 2000.3000090082746,$$

$$y_{\text{nf},2} = \text{GetY}(1200;\, D_1) \approx 800.5066386661962, \quad \Delta y_{\text{nf},2} \approx y_1 - y_{\text{nf},2} \approx 99.84343389403864,$$

$$\text{Out}_2 = 0.997 \cdot 99.84343389403864 \approx 99.54390359235653.$$

Total split output:

$$\text{Out}_{\text{split}} = \text{Out}_1 + \text{Out}_2 \approx 199.19383103212175.$$

Comparison.

$$\text{Out}_{\text{split}} - \text{Out}_{\text{outfee}}(200) \approx 199.19383103212175 - 199.19363484230544 \approx 0.00019618981631 > 0.$$

Thus the original output-fee rule is *superadditive* in this example: splitting is slightly *better* for the trader.

5.7 Contrast: Input-Fee Reinvest is Subadditive (Split Worse)

For completeness, we briefly describe the *other* common fee rule and its opposite behavior.

Definition 6 (Input-Fee Reinvest Rule)**.** Fix fee rate $f \in (0,1)$. Given state (x, y), compute D_0 from $I(x, y, D_0) = 0$. For input $\Delta > 0$:

(1) Trade only the net amount $(1-f)\Delta$ along the constant-D_0 curve:

$$y_{\text{nf}} := \text{GetY}\big(x + (1-f)\Delta;\ D_0\big).$$

(2) Credit the *full* input Δ to reserves: $x_1 := x + \Delta$.

(3) Trader receives $\text{Out}_{\text{in}}(\Delta) := y - y_{\text{nf}}$.

(4) Final state: $(x_1, y_{\text{nf}}) = (x + \Delta,\ y_{\text{nf}})$.

The key difference from the output-fee rule: here the "fee" stays in X (the mismatch $f\Delta$ between credited and traded amounts), whereas output-fee keeps fee in Y.

Proposition 5 (Subadditivity of Input-Fee Reinvest)**.** *Under the input-fee reinvest rule (Definition 6), splitting a trade into chunks is* worse *for the trader:*

$$\text{Out}_{\text{in}}(a+b;\ x_0, y_0)\ >\ \text{Out}_{\text{in}}(a;\ x_0, y_0) + \text{Out}_{\text{in}}\big(b;\ F_a^{\text{in}}(x_0, y_0)\big). \tag{36}$$

That is, the mechanism is subadditive: one-shot output strictly exceeds split output.

Sketch. After the first chunk a, the state is $(x + a, y_1)$ where $y_1 = \text{GetY}(x + (1-f)a; D_0)$. But $x + a > x + (1-f)a$, so the state $(x + a, y_1)$ is *not* on the original D_0 curve—it has moved to a higher-D curve where X is relatively more abundant. This makes subsequent $X \to Y$ trades less favorable. The "fee" injected early as extra X makes Y scarcer for subsequent chunks. □

Example 2 (Subadditivity Numbers for Input-Fee Reinvest)**.** Same parameters: $(1000, 1000)$, $A = 100$, $f = 0.003$.

- One-shot $\Delta = 200$: trade $(1-f) \cdot 200 = 199.4$ on $D_0 = 2000$ curve. $\text{Out}_{\text{in}}(200) \approx 199.1943$.
- Split 100+100: first chunk trades 99.7, second chunk trades 99.7 but on a *different* (higher-D) curve. $\text{Out}_{\text{split}} \approx 199.1942$.
- Difference: $\text{Out}_{\text{in}}(200) - \text{Out}_{\text{split}} \approx 1.4 \times 10^{-4} > 0$.

Split is worse: *subadditive*.

Remark 10 (Summary of Fee Placement Effects)**.**

Fee Rule	**Fee Stays In**	**Additivity Type**	**For Trader**
Output-fee (Curve-style)	Y reserve	Superadditive	Split better
Input-fee reinvest	X reserve	Subadditive	One-shot better
Semigroup D-update	Liquidity D	Strictly additive	Path-independent

6 (3) Target: Strict Additivity *With* Reinvested Fees on Top of PeggedSwap

6.1 Strategy: Enforce a Semigroup on D (the Natural StableSwap "Pool Size")

Because StableSwap's natural "pool size" variable is D and fee placement changes D in a path-dependent way, a clean path to strict additivity is:

> Make the D-update itself satisfy a semigroup law, then solve the invariant using the updated D.

This approach is motivated by classical results in functional equations [Aczél, 1966] and has analogues in pricing theory where semigroup structures ensure consistency of pricing operators [Garman, 1985].

6.2 Why Strict Additivity Still Matters (Even When Superadditivity Favors the Trader)

Superadditivity of the output-fee rule (Theorem 2) means splitting is strictly *better* for traders. So why pursue strict additivity at all?

(a) **Aggregator/router neutrality.** If swap outcome depends on how execution is split, routing algorithms must account for this [Angeris et al., 2022]. Strict additivity means routers can split arbitrarily without affecting final outcome.

(b) **Partial fills and TWAP.** Time-weighted execution, partial limit-order fills, and MEV-resistant chunking all benefit from path-independence.

(c) **Deterministic pool growth.** Under strict additivity with a semigroup D-update, the pool's liquidity scalar D grows deterministically with volume, independent of execution chunking.

(d) **Composability.** Protocols building on top of the AMM (vaults, strategies, aggregators) can reason about outcomes without simulating all possible split patterns [Bartoletti et al., 2024].

The superadditivity of the Curve-style rule is a *happy accident* for traders, but it still creates path-dependence that complicates integration.

6.3 D-Update Rule and the Semigroup Functional Equation

Definition 7 (D-Update Rule)**.** Let $\Gamma : \mathbb{R}_{>0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ be a rule that updates liquidity scalar D after an input of size Δ:

$$D \mapsto \Gamma(D, \Delta).$$

We require $\Gamma(D, 0) = D$ (no input means no change).

6.4 Deriving the Semigroup Condition from Strict Additivity

We now prove that the semigroup law on Γ is not an arbitrary assumption but a *necessary condition* for strict additivity.

Proposition 6 (Necessary and Sufficient Condition on Γ)**.** *Let the swap update be defined by*

$$F^{\Gamma}_{\Delta}(x,y) := \big(x+\Delta,\ \mathrm{GetY}(x+\Delta;\ \Gamma(D_0,\Delta))\big),$$

where D_0 is solved from $I(x,y,D_0)=0$.

Then strict additivity $F^{\Gamma}_{a+b} = F^{\Gamma}_{b} \circ F^{\Gamma}_{a}$ holds for all $a,b \geq 0$ ***if and only if*** *Γ satisfies*

$$\Gamma(\Gamma(D,a),b) = \Gamma(D,a+b) \qquad \forall\, D>0,\ a,b \geq 0. \tag{37}$$

Proof. Fix starting state (x,y) with $I(x,y,D_0)=0$.

One-shot path. Applying F^{Γ}_{a+b} directly:

$$F^{\Gamma}_{a+b}(x,y) = \big(x+a+b,\ \mathrm{GetY}(x+a+b;\ \Gamma(D_0,a+b))\big).$$

Two-step path. First apply F^{Γ}_{a}:

$$(x_1,y_1) := F^{\Gamma}_{a}(x,y) = \big(x+a,\ \mathrm{GetY}(x+a;\ \Gamma(D_0,a))\big).$$

Set $D_1 := \Gamma(D_0,a)$. By construction of GetY, we have $I(x_1,y_1,D_1)=0$.

Now apply F^{Γ}_{b} to (x_1,y_1). Since $I(x_1,y_1,D_1)=0$, the starting D for this step is D_1:

$$F^{\Gamma}_{b}(x_1,y_1) = \big(x_1+b,\ \mathrm{GetY}(x_1+b;\ \Gamma(D_1,b))\big) = \big(x+a+b,\ \mathrm{GetY}(x+a+b;\ \Gamma(\Gamma(D_0,a),b))\big).$$

Equality condition. The x-components match trivially ($x+a+b$). The y-components match iff the D-arguments of GetY match:

$$\mathrm{GetY}(x+a+b;\ \Gamma(D_0,a+b)) = \mathrm{GetY}(x+a+b;\ \Gamma(\Gamma(D_0,a),b)).$$

Since $\mathrm{GetY}(x';D)$ is uniquely determined by D (given x'), this holds iff

$$\Gamma(D_0,a+b) = \Gamma(\Gamma(D_0,a),b).$$

As this must hold for all starting states (hence all $D_0>0$) and all $a,b\geq 0$, we obtain (37). □

Remark 11 (The Semigroup Law is Derived, Not Assumed)**.** Equation (37) is Cauchy's functional equation in the second argument with the first argument as a parameter. Following Aczél [1966], it is the *unique* algebraic constraint making the Γ-based swap strictly additive. Any Γ violating this equation produces path-dependent outcomes.

6.5 General Solution Families for the Semigroup Equation

Proposition 7 (Standard Solution Families)**.** *Two canonical families satisfying* (37)*:*

(a) ***Additive:*** *$\Gamma(D,\Delta) = D + g(\Delta)$ where $g(a+b)=g(a)+g(b)$. If g is continuous (or monotone, or measurable), then $g(\Delta)=\kappa\Delta$ for some $\kappa\geq 0$ [Aczél, 1966].*

(b) ***Multiplicative:*** *$\Gamma(D,\Delta) = D\cdot s(\Delta)$ where $s(a+b)=s(a)s(b)$. If s is continuous, then $s(\Delta)=e^{\lambda\Delta}$ for some $\lambda\in\mathbb{R}$.*

Proof. For (a): Substituting $\Gamma(D,\Delta)=D+g(\Delta)$ into (37):

$$(D+g(a))+g(b) = D+g(a+b) \implies g(a)+g(b)=g(a+b).$$

This is Cauchy's additive functional equation. For (b), the calculation is analogous, yielding Cauchy's exponential equation. □

6.6 Strictly-Additive Fee/Reinvest Construction

Definition 8 (Strictly-Additive PeggedSwap-with-Fee on Top of StableSwap)**.** Fix $f \in (0,1)$ and choose a D-semigroup Γ (Definition 7). Given state (x, y):

(1) Compute D_0 from $I(x, y, D_0) = 0$.

(2) Update liquidity scalar:

$$D_1 := \Gamma(D_0, \Delta).$$

(3) Set $x_1 := x + \Delta$.

(4) Define y_1 as the unique solution of

$$I(x_1, y_1, D_1) = 0 \quad (\text{equivalently } y_1 = \mathrm{GetY}(x_1; D_1)).$$

Define

$$F^{\Gamma}_{\Delta}(x, y) := (x_1, y_1), \qquad \mathrm{Out}_{\Gamma}(\Delta; x, y) := y - y_1.$$

Theorem 3 (Strict Additivity of the Construction)**.** *If Γ satisfies* (37)*, then the mechanism in Definition 8 is strictly additive:*

$$\mathrm{Out}_{\Gamma}(a + b; x, y) = \mathrm{Out}_{\Gamma}(a; x, y) + \mathrm{Out}_{\Gamma}\big(b; F^{\Gamma}_{a}(x, y)\big)$$

for all $a, b \geq 0$. Equivalently, $F^{\Gamma}_{a+b} = F^{\Gamma}_{b} \circ F^{\Gamma}_{a}$.

Proof with Explicit Intermediate Steps. Fix starting state (x, y) and let D_0 solve $I(x, y, D_0) = 0$.

Step 1 (One-shot). For $\Delta = a + b$, the construction gives

$$D_{\text{one}} = \Gamma(D_0, a + b), \quad x_{\text{one}} = x + a + b, \quad y_{\text{one}} = \mathrm{GetY}(x + a + b;\, D_{\text{one}}).$$

Step 2 (Two-step). After first step a:

$$D_1 = \Gamma(D_0, a), \quad x_1 = x + a, \quad y_1 = \mathrm{GetY}(x + a;\, D_1).$$

After second step b:

$$D_2 = \Gamma(D_1, b) = \Gamma(\Gamma(D_0, a), b), \quad x_2 = x_1 + b = x + a + b, \quad y_2 = \mathrm{GetY}(x + a + b;\, D_2).$$

Step 3 (Use the semigroup law). By (37),

$$D_2 = \Gamma(\Gamma(D_0, a), b) = \Gamma(D_0, a + b) = D_{\text{one}}.$$

Step 4 (Conclude equality). Since $x_2 = x_{\text{one}}$ and $D_2 = D_{\text{one}}$, the defining equation $I(x, y, D) = 0$ yields the same unique y:

$$y_2 = \mathrm{GetY}(x + a + b;\, D_2) = \mathrm{GetY}(x + a + b;\, D_{\text{one}}) = y_{\text{one}}.$$

Hence the final states match and $F^{\Gamma}_{a+b} = F^{\Gamma}_{b} \circ F^{\Gamma}_{a}$, establishing strict additivity. □

6.7 Concrete Choice That Matches "Fee Reinvest" Intuition: Additive-D

For a pegged pair normalized to price 1, choose

$$\Gamma(D, \Delta) = D + f\Delta.$$

Then each trade increases D deterministically by the "fee notional".

Example 3 (Numbers: Strict Additivity Achieved)**.** Take $(x_0, y_0) = (1000, 1000)$, $A = 100$, $f = 0.003$, and $\Gamma(D, \Delta) = D + f\Delta$. At balance $D_0 = 2000$.

One-shot $\Delta = 200$:

$$D_1 = 2000 + 0.003 \cdot 200 = 2000.6, \quad x_1 = 1200, \quad y_1 = \mathrm{GetY}(1200;\, 2000.6) \approx 800.8062736266,$$

$$\mathrm{Out}_\Gamma(200) = 1000 - 800.8062736266 \approx 199.1937263734.$$

Split $100 + 100$:
First 100:

$$D = 2000.3,\ x = 1100,\ y \approx 900.3500635568,\ \mathrm{Out} \approx 99.6499364432,$$

Second 100:

$$D = 2000.6,\ x = 1200,\ y \approx 800.8062736266,\ \mathrm{Out} \approx 99.5437899302.$$

Total:

$$99.6499364432 + 99.5437899302 = 199.1937263734,$$

matching the one-shot result exactly (up to numerical precision).

7 Discussion: What "Pool Growth" Means and Practical Implications

7.1 Why D is the Right Notion of "Pool Size"

A single $X \to Y$ swap cannot make *both* reserves increase simultaneously—Y is paid out to the trader. So "pool growth" from fees cannot mean "both reserves go up after every trade."

The correct notion of growth for StableSwap-type pools is *increase in the liquidity scalar* D. Recall that D scales linearly with reserves when the pool is balanced: if $(x, y) \mapsto (tx, ty)$, then $D \mapsto tD$. Thus D captures the "total liquidity" in a price-independent way, analogous to the value function characterizations in Milionis et al. [2022].

7.2 How the Semigroup Construction Achieves Deterministic Pool Growth

Under the strictly-additive construction (Definition 8) with $\Gamma(D, \Delta) = D + f\Delta$:

- Each swap increases D by exactly $f\Delta$, regardless of how execution is split.
- After arbitrage restores balance, the increased D manifests as higher reserves on *both* sides.
- Pool growth is deterministic and proportional to traded volume.

This is in contrast to both the output-fee and input-fee rules, where the effective D increase depends on the execution path.

7.3 Practical Implications

(1) **For traders:** Strict additivity means outcome depends only on total input, not on chunking. No gaming of split patterns.

(2) **For aggregators:** Routing can split trades arbitrarily without affecting final output. This simplifies optimization problems of the type studied by Angeris et al. [2022].

(3) **For LPs:** Pool growth is deterministic. Fee revenue can be predicted from volume alone, improving upon the stochastic fee accrual analyzed by Milionis et al. [2022].

(4) **For protocols:** Composability is cleaner when swap outcomes are path-independent [Bartoletti et al., 2024].

8 Summary

- **Fee-free StableSwap is strictly additive** (one-shot = split) because swaps are motion on a constant-D curve (Theorem 1).
- The **original Curve-style output-fee rule is superadditive** (split > one-shot, better for trader) under a natural monotonicity condition (Theorem 2).
- The **input-fee reinvest rule is subadditive** (split < one-shot, worse for trader)—see Proposition 5.
- The **semigroup condition on Γ is derived**, not assumed: it is the unique constraint making the D-update swap strictly additive (Proposition 6).
- To **achieve strict additivity with reinvested fees**, enforce the semigroup law on D and solve the invariant on the updated D (Theorem 3); additive-D ($\Gamma(D, \Delta) = D + f\Delta$) is the simplest concrete choice.

Mechanism	**Additivity**	**Comparison**	**Path-Dependent?**
Fee-free StableSwap	Strictly additive	$\text{Out}(a+b) = \text{Out}_{\text{split}}$	No
Output-fee (Curve)	Superadditive	$\text{Out}(a+b) < \text{Out}_{\text{split}}$	Yes
Input-fee reinvest	Subadditive	$\text{Out}(a+b) > \text{Out}_{\text{split}}$	Yes
Semigroup D-update	Strictly additive	$\text{Out}(a+b) = \text{Out}_{\text{split}}$	No

References

J. Aczél. *Lectures on Functional Equations and Their Applications*. Academic Press, New York, 1966.

J. Aczél and J. Dhombres. *Functional Equations in Several Variables*. Encyclopedia of Mathematics and its Applications, vol. 31. Cambridge University Press, 1989.

G. Angeris and T. Chitra. Improved price oracles: Constant function market makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT '20)*, pages 80–91, 2020. arXiv:2003.10001.

G. Angeris, A. Agrawal, A. Evans, T. Chitra, and S. Boyd. Constant function market makers: Multi-asset trades via convex optimization. *Springer*, 2022. arXiv:2107.12484.

G. Angeris, T. Chitra, A. Evans, and S. Boyd. Optimal routing for constant function market makers. *arXiv preprint arXiv:2204.05238*, 2022.

G. Angeris, T. Chitra, G. Diamandis, A. Evans, and K. Kulkarni. The geometry of constant function market makers. In *Proceedings of the 24th ACM Conference on Economics and Computation (EC '24)*, 2024. arXiv:2308.08066.

A. Baggiani, M. Herdegen, and A. Sánchez-Betancourt. Optimal dynamic fees in automated market makers. *arXiv preprint arXiv:2506.02869*, 2025.

M. Bartoletti, R. Marchesin, and R. Zunino. DeFi composability as MEV non-interference. In *Financial Cryptography and Data Security (FC'24)*, 2024.

M. Bichuch and Z. Feinstein. Axioms for automated market makers: A mathematical framework in FinTech and decentralized finance. *Operations Research*, 2025. arXiv:2210.01227.

M. Egorov. StableSwap – efficient mechanism for stablecoin liquidity. Curve Finance Whitepaper, November 2019. `https://curve.fi/files/stableswap-paper.pdf`.

M. Egorov. Automatic market-making with dynamic peg. Curve Finance Whitepaper, June 2021. `https://docs.curve.finance/assets/pdf/whitepaper_cryptoswap.pdf`.

A. Evans, G. Angeris, and T. Chitra. Optimal fees for geometric mean market makers. In *FC 2021 Workshops*, 2021. arXiv:2104.00446.

R. Frongillo, D. Papireddygari, and B. Waggoner. An axiomatic characterization of CFMMs and equivalence to prediction markets. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, LIPIcs Vol. 287, 2024.

M. Garman. Towards a semigroup pricing theory. *The Journal of Finance*, 40(3):847–862, 1985.

E. Hertzog, G. Benartzi, and G. Benartzi. Bancor protocol: A protocol for intrinsically tradeable cryptographic tokens. Bancor Whitepaper, 2017 (revised 2018).

F. Martinelli and N. Mushegian. Balancer: A non-custodial portfolio manager, liquidity provider, and price sensor. Balancer Whitepaper, September 2019. `https://docs.balancer.fi/whitepaper.pdf`.

J. Milionis, C. C. Moallemi, T. Roughgarden, and A. L. Zhang. Automated market making and loss-versus-rebalancing. In *Science of Blockchain Conference (SBC'22)*, 2022. arXiv:2208.06046.

J. C. Schlegel, M. Kwaśnicki, and A. Mamageishvili. Axioms for constant function market makers. In *Proceedings of the 24th ACM Conference on Economics and Computation (EC '23)*, 2023. arXiv:2210.00048.

L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, Annals of Mathematics Studies, vol. 28, pages 307–317. Princeton University Press, 1953.

W. W. Sharkey. *The Theory of Natural Monopoly*. Cambridge University Press, 1982.