StableSwap Portfolio Value:
From Invariant to Numerical Solver

Problem Statement

Given the StableSwap invariant D (computed by `newton_D`), a known marginal price $p = dx_0/dx_1$ (computed by `get_p`), and the amplification parameter `_amp`, find the **portfolio value**:

$$V = x_0 + p \cdot x_1$$

where the balances (x_0, x_1) are **not explicitly known** but are implicitly defined by (D, p) through the invariant and price conditions.

This quantity V represents the value of the pool's reserves measured at the current marginal price—a key input for impermanent loss calculations, oracle design, and risk metrics.

1 The StableSwap Invariant

1.1 Whitepaper formulation

The StableSwap invariant for n coins (from the Egorov 2019 whitepaper) is:

$$An^n \sum x_i + D = ADn^n + \frac{D^{n+1}}{n^n \prod x_i}$$

For $n = 2$ coins, defining $\alpha = An^n = 4A$ (called `Ann` in the code, equal to `2 * _amp`):

$$\alpha(x_0 + x_1) + D = \alpha D + \frac{D^3}{4\, x_0 x_1}$$

1.2 Code conventions

In `StableswapMath.vy`, the amplification parameter `_amp` includes an `A_MULTIPLIER = 10000` scaling factor for integer precision:

Symbol	Code variable	Relation
`_amp`	`_amp`	$= A \cdot N^{N-1} \cdot$ `A_MULTIPLIER`
α	`Ann`	$=$ `_amp` $\cdot N$
Effective An^n		$=$ `Ann / A_MULTIPLIER`

For $N = 2$ and $A = 100$: `_amp = 100 * 2 * 10000 = 2,000,000`.

2 The Price Function

The marginal price $p = -dx_0/dx_1 > 0$ along the invariant curve (constant D) is computed by `get_p` as:

$$p = \frac{x_0(\alpha\, x_1 + q)}{x_1(\alpha\, x_0 + q)}, \qquad q = \frac{D^3}{4\, x_0 x_1}$$

This follows from implicit differentiation of the invariant:

$$p = \frac{F_{x_1}}{F_{x_0}} = \frac{\alpha + D^3/(4x_0x_1^2)}{\alpha + D^3/(4x_0^2x_1)}$$

3 Analytical Derivation

3.1 Cross-multiplying the price equation

Starting from $p = \frac{x_0(\alpha x_1+q)}{x_1(\alpha x_0+q)}$ and cross-multiplying:

$$4\alpha(x_0x_1)^2(p-1) = D^3(x_0 - p\,x_1) \tag{$*$}$$

3.2 The key algebraic identity

Define $V = x_0 + p\,x_1$ (the target) and $Q = x_0 - p\,x_1$. Then:

$$V^2 = Q^2 + 4p\,x_0x_1$$

3.3 Expressing in terms of Γ

Define $\Gamma = \frac{D^3}{4\,x_0x_1}$, which by the invariant equals $\alpha(S-D)+D$ where $S = x_0 + x_1$. Then $x_0x_1 = D^3/(4\Gamma)$ and from ($*$):

$$Q = \frac{\texttt{\_amp}\,(p-1)\,D^3}{2\,\Gamma^2}$$

Substituting into the identity:

$$\boxed{V^2 = \frac{p\,D^3}{\Gamma} + \frac{\texttt{\_amp}^2\,(p-1)^2\,D^6}{4\,\Gamma^4}} \tag{1}$$

3.4 Second relation

From $S = x_0 + x_1$ and $V = x_0 + p\,x_1$, we can express $x_0 = \frac{pS-V}{p-1}$ and $x_1 = \frac{V-S}{p-1}$, giving:

$$(1+p)\,V = 2pS - (p-1)\,Q$$

Substituting Q and $S = [\Gamma + (2\,\texttt{\_amp} - 1)D]/(2\,\texttt{\_amp})$:

$$\boxed{V = \frac{p[\Gamma + (2\,\texttt{\_amp}-1)D]}{\texttt{\_amp}\,(1+p)} - \frac{\texttt{\_amp}\,(p-1)^2\,D^3}{2(1+p)\,\Gamma^2}} \tag{2}$$

3.5 No closed-form solution

The two boxed equations jointly determine V and Γ. Eliminating V yields a **degree-6 polynomial** in Γ:

$$16p(\Gamma+(\alpha-1)D)^2\Gamma^4 - 4\alpha^2(p-1)^2D^3(\Gamma+(\alpha-1)D)\Gamma^2 - 4\alpha^2(1+p)^2D^3\Gamma^3 - \alpha^4(p-1)^2D^6 = 0$$

Since degree-6 polynomials have no general radical solution, **there is no simple closed-form** for V in terms of $(D, p, \texttt{\_amp})$.

3.6 Envelope theorem

A clean integral representation follows from the envelope theorem. Since $V(p) = x_0(p) + p\,x_1(p)$:

$$\frac{dV}{dp} = \frac{dx_0}{dp} + x_1 + p\frac{dx_1}{dp} = x_1$$

(because $dx_0 + p\,dx_1 = 0$ on the invariant curve). Therefore:

$$V(p) = D + \int_1^p x_1(s)\,ds$$

where $V(1) = D$ since $x_0 = x_1 = D/2$ when $p = 1$.

3.7 Limiting cases

Regime	Formula	Note
$p = 1$ (balanced)	$V = D$	Exact for all A
`_amp` $\to 0$ (constant-product)	$V = D\sqrt{p}$	Recovers Uniswap
`_amp` $\to \infty$ (constant-price)	$V = D$	Only $p = 1$ is reachable

4 Numerical Solver Design

Since no closed-form exists, we solve the problem numerically. The solver finds x_1 such that the price computed from the invariant matches the target p.

4.1 Problem reduction

The observation is that **price is monotonically decreasing in x_1**: increasing x_1 (more of coin 1) makes coin 0 relatively scarcer, so the exchange rate $p = dx_0/dx_1$ decreases. This guarantees a unique solution and enables bisection.

For any candidate x_1:

1. Compute $x_0 =$ `get_y(_amp,` $[0, x_1]$, D, 0`)`—Newton's method on the invariant
2. Compute $p_{\text{cur}} =$ `get_p(`$[x_0, x_1]$, D, `[_amp,`0`])`
3. Compare p_{cur} with the target p

4.2 Algorithm: Bisection + Newton hybrid

Phase 1—Bisection (10 iterations):

Starting from the full bracket $x_1 \in [1, D-1]$, bisect to reduce the interval by $2^{10} = 1024\times$. This brings the estimate within $\sim$0.1% of the true value regardless of how imbalanced the pool is.

Phase 2—Newton's method (3–4 iterations):
From the bisection midpoint, apply Newton's method:

$$x_1^{(k+1)} = x_1^{(k)} - \frac{p_{\text{cur}} - p}{dp/dx_1}$$

The derivative dp/dx_1 is evaluated numerically:

$$\frac{dp}{dx_1} \approx \frac{\texttt{get\_p}(x_1 + h) - \texttt{get\_p}(x_1)}{h}, \qquad h = \max(x_1/10^7,\ 1)$$

Newton updates are clamped to the bisection bracket to prevent divergence.

4.3 Implementation details

All arithmetic uses **wad integers** ($1.0 = 10^{18}$), matching the on-chain Vyper representation. The ported functions (`newton_D`, `get_y`, `get_p`) reproduce the Vyper code's integer division semantics exactly.

The final value is computed as:

$$V = x_0 + p \cdot x_1 \mathbin{//} 10^{18}$$

where // denotes integer (floor) division.

5 Validation

5.1 Test methodology

For each test case:

1. Start with **known balances** (x_0, x_1) and amplification `_amp`
2. Compute $D =$ `newton_D(_amp,` $[x_0, x_1]$`)`
3. Compute $p =$ `get_p(`$[x_0, x_1],\ D,\ [$`_amp`$, 0]$`)`
4. Compute $V_{\text{expected}} = x_0 + p \cdot x_1 \mathbin{//} 10^{18}$ directly
5. Compute $V_{\text{solver}} =$ `portfolio_value(`$D, p,$ `_amp)` using only $(D, p,$ `_amp`$)$
6. Compare: $|V_{\text{solver}} - V_{\text{expected}}| \leq \varepsilon$

5.2 Test cases

Eleven test cases cover the parameter space:

Case	x_0	x_1	A	Description
1–3	1.0	1.0	5, 100, 1000	Balanced pools
4–5	1.2 / 0.8	0.8 / 1.2	100	Mild imbalance, $p \gtrless 1$
6	1.1	0.9	1000	Mild imbalance, high A
7–8	1.5 / 0.5	0.5 / 1.5	100	Moderate imbalance
9–10	2.0 / 0.5	0.5 / 2.0	5	Heavy imbalance, low A
11	3.0	0.333	5	Very heavy imbalance

5.3 Results

```
Case                        D            p      V_expected       V_computed    err  it
-------------------------------------------------------------------------------------
balanced, A=100         2.000000      1.000000  2.0000000000    2.0000000000   0    0
balanced, A=5           2.000000      1.000000  2.0000000000    2.0000000000   0    0
balanced, A=1000        2.000000      1.000000  2.0000000000    2.0000000000   0    0
mild p>1, A=100         1.999793      1.002160  2.0017280735    2.0017280735   0   13
mild p<1, A=100         1.999793      0.997845  1.9974134769    1.9974134769   0   13
mild p>1, A=1000        1.999995      1.000102  2.0000917845    2.0000917845   2   13
moderate p>1, A=100     1.998346      1.008828  2.0044138566    2.0044138566   1   13
moderate p<1, A=100     1.998346      0.991250  1.9868743002    1.9868743002   1   13
heavy p>1, A=5          2.440354      1.249805  2.6249027116    2.6249027116   2   13
heavy p<1, A=5          2.440354      0.800125  2.1002490970    2.1002490970   1   13
very heavy, A=5         3.112156      1.892649  3.6308828680    3.6308828680   1   14
```

All 11 tests pass. Maximum error: **2 wei** (2 parts in 10^{18}, i.e. $< 10^{-17}$ relative error). Convergence: **13–14 iterations** (10 bisection + 3–4 Newton).

5.4 Why the error is so small

The portfolio value $V = x_0 + p\,x_1$ is **first-order insensitive** to errors in x_1. If the solver's x_1 is off by δ, then x_0 shifts by approximately $-p\,\delta$ (since $dx_0/dx_1 = -p$ on the invariant curve), and:

$$\Delta V \approx (-p\,\delta) + p\,\delta = 0$$

The error in V is therefore **second-order** in δ—this is a direct consequence of V being the value functional whose gradient is tangent to the invariant curve at price p.

6 Summary

Item	Result
Closed-form for $V(D, p, A)$?	**No**—requires solving a degree-6 polynomial
Integral representation	$V(p) = D + \int_1^p x_1(s)\,ds$ (envelope theorem)
Limiting cases	$V = D$ (balanced), $V = D\sqrt{p}$ (constant-product)
Numerical solver	Bisection + Newton on x_1, 13–14 iterations
Accuracy	≤ 2 wei ($< 10^{-17}$ relative)
Implementation	Python, wad integers (10^{18}), matching Vyper arithmetic

Files produced

File	Contents
`portfolio_value.md`	Mathematical derivation (Markdown)
`portfolio_value.pdf`	Mathematical derivation (PDF, LaTeX-typeset)
`portfolio_value_solver.py`	Python solver + validation tests
`report.md`	This report (Markdown)
`report.pdf`	This report (PDF)