Pricing StableSwap pools

Sergio A. Yuhjtman, Balancer Labs

Abstract

We determine the total value in a StableSwap pool as a function of the invariant and the prices of the constituent tokens. The solution combines analytical and numerical methods to enable efficient implementation in a smart contract.

1 Introduction and problem statement

Consider a *constant function automated market maker* (see [2]) such as a *constant product AMM*([1], [5]), or a *StableSwap AMM*[3]. The pool issues its own pool-token to represent liquidity positions. This pool-token may be lent or borrowed through a lending application, but its price must be understood precisely. At a given instant, if $(x_i)_{1\leq i\leq n}$ are the balances of the pool components and S is the pool-token supply, we can express the price as:

$$P_{pt} = \frac{1}{S}\sum_{i=1}^{n} p_i x_i$$

where p_i is the price of the i-th token expressed in the desired unit. In words, this is the total value of the pool divided by the pool-token supply.

Let us call $\mathbb{R}_+^n \xrightarrow{f} \mathbb{R}_+$ the defining function for the AMM. This means that the equation

$$f(x) = D$$

defines the swap rule at zero fee, for each constant $D > 0$. As before, the vector x represents the pool reserves. The function f is continuously differentiable, strictly increasing in each variable and satisfies $\lim_{x_i\to 0} f(x) = 0$, $\lim_{x_i\to\infty} f(x) = +\infty$. Thus, the equation globally defines the value for each reserve x_i as a function of the remaining values. From the formula for the partial derivative of an implicit function, it follows that the prices of the tokens determined by the pool state are given by the partial derivatives of f. For instance, the price of token 2 in units of token 1 is $\frac{\partial_2 f}{\partial_1 f}$. This is the infinitesimal amount of token 2 that the pool exchanges for each unit of token 1. Thus, the gradient of f expresses the internal prices of the tokens in some unit. Unfortunately, for the desired use case it is not satisfactory to use the formula $\langle \nabla f(x), x\rangle$ to express the total value of the pool, because it is possible for an agent to manipulate the state of the pool during a short period (see for instance [4]). However, nobody can manipulate the invariant D. By obtaining the prices $p \in \mathbb{R}_+^n$ from an oracle, if we can find $x \in \mathbb{R}_+^n$ and $k \in \mathbb{R}_+$ such that

$$\begin{cases} \nabla f(x) = kp \\ f(x) = D \end{cases} \tag{1}$$

then x is the non-manipulated vector of pool balances. We can use them to compute the total value held by the pool and the price of the pool token.

In this note, we address this system of equations for the case of StableSwap AMM. First, we reduce the system to a single equation and locate the unique relevant root. We then show the convergence of Newton's method for finding this root, starting from a suitable point.

A theoretical proof of convergence to the correct root is particularly valuable, as it rules out the possibility of hidden problematic cases that could result in financial losses.

1.1 Weighted pools

As a simple but important example, let us derive the solution to (1) for the case of a weighted pool (see [5]). Here the defining function is

$$f(x) = \prod_{i=1}^{n} x_i^{w_i}$$

for some weights $w_i > 0$ such that $\sum_{i=1}^n w_i = 1$. Assuming the identities (1),

$$kp_i = \partial_i f(x) = w_i x_i^{-1} f(x) = w_i x_i^{-1} D$$

$$x_i = \frac{Dw_i}{kp_i}$$

$$k = \frac{Dw_i}{x_i p_i}$$

By raising each side of the last equation to the power of w_i and multiplying them together, we obtain:

$$k = \prod_{j=1}^n \left(\frac{w_j}{p_j}\right)^{w_j}$$

leading us to the expressions

$$x_i = \frac{Dw_i}{p_i} \prod_{j=1}^n \left(\frac{p_j}{w_j}\right)^{w_j}$$

$$\sum_{i=1}^n x_i p_i = D \prod_{j=1}^n \left(\frac{p_j}{w_j}\right)^{w_j}$$

1.2 Lower and upper prices

For a general *constant function automated market maker*, there is a very efficient method to compute the price such that, under a pool manipulation, the result can deviate only to one side. This might be useful in applications, so here we give the definitions and the corresponding inequalities.

Recall that f is continuously differentiable, strictly increasing in each variable and satisfies $\lim_{x_i \to 0} f(x) = 0$, $\lim_{x_i \to \infty} f(x) = +\infty$. Assume, in addition, that f is strictly convex.

Let $p \in \mathbb{R}_+^n$ be a vector of prices, $x \in \mathbb{R}_+^n$ the state of the pool, and denote $D = f(x)$.

By a compactness argument, there is a $\hat{x} \in f^{-1}(D)$ that minimizes $\sum_i p_i \hat{x}_i$ on $f^{-1}(D)$. Using the strict convexity and monotonicity of f, it follows that this minimum is unique. Since the hypersurface $f^{-1}(D)$ has no boundary, the Lagrange multipliers condition $\nabla f(\hat{x}) = kp$ holds for some $k > 0$. Conversely, starting with $x \in \mathbb{R}_+^d$, the strict convexity implies that the expression $\langle \nabla f(x), - \rangle$ on $f^{-1}(D)$ is minimized at x. This shows the existence and uniqueness of solution for the system 1. The value of the pool is $V := \sum_i p_i \hat{x}_i$.

We can define the upper value

$$V_+ := \sum_i p_i x_i \geq \sum_i p_i \hat{x}_i = V$$

and use it to define the upper price of the pool-token. On the other hand, we have the lower value

$$V_- := \left(\sum_i \partial_i f(x).x_i\right) . \min_i \left\{\frac{p_i}{\partial_i f(x)}\right\}$$

$$V_- \leq \left(\sum_i \partial_i f(x).\hat{x}_i\right) . \min_i \left\{\frac{p_i}{\partial_i f(x)}\right\} \leq \sum_i p_i \hat{x}_i = V$$

2 StableSwap function

The StableSwap pool type was first introduced by Michael Egorov in [3] and was first implemented by the Curve protocol. The function f is given implicitly by the equation

$$An^n \sum_{i=1}^n x_i + D = ADn^n + \frac{D^{n+1}}{n^n \prod_{i=1}^n x_i}$$

for $n \geq 2$. Equivalently, we can write this equation as

$$F(x_1, ..., x_n, D) = 0$$

where

$$\mathbb{R}_+^{n+1} \xrightarrow{F} \mathbb{R}$$

$$F(x_1, ..., x_n, D) = a(\sum_{i=1}^n x_i)(\prod_{i=1}^n x_i) - bD \prod_{i=1}^n x_i - D^{n+1}$$

for $a = An^{2n}$, $b = n^n(An^n - 1) = a - n^n$. It is convenient to assume $b > 0$, or equivalently $An^n > 1$, a condition that holds in known blockchain applications. Note that we adopt the symbols a and b for notational convenience, even though the function is determined by a single parameter A.

2.1 Basic properties of F and the implicit functions

In this subsection we concisely show that the expression $F = 0$ defines a *constant function automated market maker* for a function f with the properties required for the general case in the introduction.

Since $F(x, D)$ is strictly decreasing in D and we can directly verify $F(x, 0) > 0$, $F(x, D) \underset{D\to\infty}{\longrightarrow} -\infty$, we conclude that for each x there is a unique $D > 0$ that satisfies $F = 0$. Thus, we may write $D = f(x)$. Rather than using the symbol f, we follow the customary notational abuse of identifying variables with implicit functions.

We compute the partial derivatives of F:

$$\partial_{x_j} F(x, D) = a(\prod_i x_i) + a(\sum_i x_i)(\prod_{i\neq j} x_i) - bD \prod_{i\neq j} x_i \tag{2}$$

Proposition 2.1. *With $\mathbb{R}^n_+ \xrightarrow{D} \mathbb{R}$ defined as above, it holds*
(a) $D \leq \sum_i x_i$. The identity holds if and only if $x_i = x_j$ $\forall i, j$.
(b) $D \in \mathcal{C}^\infty$, and it is strictly increasing in each variable.
(c) $D \underset{x_i\to 0}{\longrightarrow} 0$ and $D \underset{x_i\to\infty}{\longrightarrow} +\infty$.

Proof.
(a) By calling $S = \sum_i x_i$ and $\Pi = \prod_i x_i$, it suffices to show that $F(x, S) \leq 0$. This is equivalent to

$$aS\Pi - bS\Pi - S^{n+1} \leq 0$$
$$\Pi^{1/n} \leq S/n$$

which is the inequality between arithmetic and geometric means. We have used $a - b = n^n$.

(b) The fact $D \in \mathcal{C}^\infty$ is inherited from the infinite differentiability of F. By the formula for the derivative of an implicit function, we have

$$\partial_{x_j} D = -\frac{\partial_{x_j} F}{\partial_D F}$$

where

$$\partial_D F(x, D) = -b\Pi - (n + 1)D^n < 0$$

and

$$\partial_{x_j} F(x, D) = a\Pi + (aS - bD) \prod_{i\neq j} x_i > 0$$

because from (a) it follows $aS - bD \geq (a - b)S > 0$. This shows that $\partial_{x_j} D > 0$.

(c) For the limit $D \underset{x_i\to 0}{\longrightarrow} 0$, we need to show that for any $D_0 > 0$ (no matter how small) there is always a value for x_i (leaving fixed x_j for $j \neq i$) such that $F(x, D_0) < 0$. This is easy to verify.

For the limit $D \underset{x_i\to\infty}{\longrightarrow} +\infty$, we need to show that for any $D_0 > 0$ (no matter how big) there is always a value for x_i (leaving fixed the remaining variables) such that $F(x, D_0) > 0$. This follows by using again $aS - bD \geq (a - b)S = n^n S$. □

For more background on the StableSwap functions the reader is referred to [6]. As shown in Subsection 1.2, the existence and uniqueness of a solution to system (1) can be ensured by the strict convexity of D, i.e., the positive definiteness of its Hessian matrix. However, this step is optional, as our construction leads explicitly to the unique solution.

3 Reduction of the system

Now that we have a good understanding of the StableSwap function, we turn to solving the system (1). First we solve the equation for the gradient and obtain a closed expression for the balances x as a function of a parameter k. The correct value for k will be the one that makes x satisfy $f(x) = D$, or equivalently $F(x, D) = 0$.

3.1 Solution to the equation for the gradient

The gradient of f, now also denoted ∇D, is equal to $\frac{1}{-\partial_D F}\nabla_x F$, so we may simply replace ∇f by $\nabla_x F$. By (2), the equation for the gradient can be written coordinate-wise as:

$$a + \frac{a\sum x_i - bD}{x_j} = \frac{\tilde{k}}{\prod x_i} p_j$$

for some $\tilde{k} > 0$. By calling $k = \frac{\tilde{k}}{\prod x_i}$ we can write the equations as

$$a\sum_{i=1}^{n} x_i - bD + (a - kp_j)x_j = 0 \quad \forall j \in \{1, ..., n\}$$

or equivalently

$$-\sum_{i=1}^{n} x_i + cD + (kr_j - 1)x_j = 0 \quad \forall j \in \{1, ..., n\} \tag{3}$$

where

$$r_i = p_i/a, \quad c = b/a$$

Notice that the system (3) encodes the condition for the gradient but not the condition for the invariant $f(x) = D$. By treating k as a constant, this system is linear and there is a simple direct solution.

Equalling left hand sides between (3) for $j = 1$ and generic j we obtain

$$(kr_j - 1)x_j = (kr_1 - 1)x_1$$

$$x_j = \frac{kr_1 - 1}{kr_j - 1}x_1 \tag{4}$$

Here it is easy to rule out the case $kr_j - 1 = 0$. Moreover, in subsection 3.3 we will independently show that under $F(x, D) = 0$, it holds $kr_j - 1 > 0$ for every j. Using these values into the equation for $j = 1$ we reach

$$(kr_1 - 1)\left((\sum_{i=1}^{n} \frac{1}{kr_i - 1}) - 1\right)x_1 = cD$$

$$x_1 = \frac{cD}{(kr_1 - 1)}\left((\sum_{i=1}^{n} \frac{1}{kr_i - 1}) - 1\right)^{-1}$$

Using again (4) we can extend this to a formula for all the variables

$$x_j = \frac{cD}{(kr_j - 1)}\left((\sum_{i=1}^{n} \frac{1}{kr_i - 1}) - 1\right)^{-1}$$

3.2 Consistency with D

We must find the value of k that is consistent with the given D. To do so, we plug our values for x_j into the invariant equation. Calling $T = (\sum_{i=1}^{n} \frac{1}{kr_i - 1}) - 1$ we have

$$x_j = \frac{cDT^{-1}}{kr_i - 1} \tag{5}$$

We use these values into

$$a(\sum_{i=1}^{n} x_i)(\prod_{i=1}^{n} x_i) - bD\prod_{i=1}^{n} x_i - D^{n+1} = 0$$

Notice that D cancels out. We put $P = \prod_{i=1}^{n}(kr_i - 1)$

$$ac^{n+1}T^{-(n+1)}(T + 1)P^{-1} - bc^nT^{-n}P^{-1} - 1 = 0$$

$$T^{-n}(1 + T^{-1})P^{-1} - T^{-n}P^{-1} - \frac{1}{ac^{n+1}} = 0$$

$$T^{-(n+1)}P^{-1} - \frac{1}{ac^{n+1}} = 0$$

$$\boxed{T^{n+1}P = \alpha} \tag{6}$$

for $\alpha = ac^{n+1}$. This last expression (6) is our equation for k. Recall that

$$T(k) = (\sum_{i=1}^{n} \frac{1}{kr_i - 1}) - 1$$

$$P(k) = \prod_{i=1}^{n}(kr_i - 1)$$

We will apply Newton's method to solve (6).

3.3 Signs of $kr_i - 1$ and $T(k)$ at a solution

Rewrite the system (3) as:

$$\sum_{i=1}^{n} x_i = \frac{b}{a} D + (kr_j - 1)x_j$$

By replacing this value for the sum into the invariant equation we get

$$(kr_j - 1)x_j \prod_{i=1}^{n} x_i = \frac{D^{n+1}}{a}$$

Since $x_j, D, a > 0$, we conclude that $kr_j - 1 > 0$ for every j. Combine the last expression and (5) to obtain

$$x_j = \frac{D^{n+1}}{a(\prod x_i)(kr_j - 1)} = \frac{cDT^{-1}}{kr_j - 1}$$

from where we can extract the identity

$$T = bD^{-n} \prod x_i > 0$$

As we will see next, these signs are crucial for finding the correct value of k.

3.4 Existence, uniqueness and location of solution

The condition $kr_i - 1 > 0$ can be expressed as $k > \rho$, where $\rho = \frac{1}{\min r_i}$. Since k lies in this region, we do not face the singularities in the expression for T. Besides, $T(k)$ is strictly decreasing and its limit values are $\lim_{\rho^+} T = +\infty$, $\lim_{+\infty} T = -1$. Therefore, T has a unique root σ in the interval $(\rho, +\infty)$. Since we are only interested in the region $T > 0$, we restrict our study to the interval $(\rho, \sigma] \subset \mathbb{R}$.

Let us call

$$(\rho, \sigma] \xrightarrow{G} \mathbb{R}, \quad G(k) = T(k)^{n+1} P(k) - \alpha$$

The equation for k (6) takes the form $G = 0$. Observe that

$$\lim_{k \to \rho^+} G(k) = +\infty, \quad G(\sigma) < 0,$$

whence there is a solution to $G(k) = 0$ for $k \in (\rho, \sigma)$. To prove uniqueness, we will show that $G' < 0$ in (ρ, σ).

The following lemma will be needed also in the next section.

Lemma 3.1. *Let $I \xrightarrow{\varphi, \psi} \mathbb{R}$ be two decreasing functions defined on $I \subset \mathbb{R}$, and let $r_1, ..., r_n \in \mathbb{R}$. It holds the inequality*

$$n \sum_{i=1}^{n} \varphi(r_i)\psi(r_i) \geq \left(\sum_{i=1}^{n} \varphi(r_i) \right) \left(\sum_{i=1}^{n} \psi(r_i) \right)$$

Proof. For any pair i, j we have

$$(\varphi(r_i) - \varphi(r_j))(\psi(r_i) - \psi(r_j)) \geq 0$$

The lemma follows from summing the inequalities over all pairs. □

Proposition 3.2. *The function $(\rho, \sigma] \xrightarrow{G} \mathbb{R}$ is strictly decreasing.*

Proof. We first compute the derivative of G.

$$G = T^{n+1} P - \alpha$$

$$G' = (n+1) T^n T' P + T^{n+1} P'$$

$$G' = T^n [(n+1) T' P + T P']$$

$$T'(k) = -\sum_i r_i (kr_i - 1)^{-2}$$

$$P'(k) = P \sum_i \frac{r_i}{kr_i - 1}$$

$$G' = T^n P \left((n+1) T' + T \sum_i \frac{r_i}{kr_i - 1} \right)$$

We know $T^n P \geq 0$, taking the value 0 only at σ. The third factor is always negative, as we will show next. Let us call H this factor.

$$H < nT' + \sum_i \frac{1}{kr_i - 1} \sum_i \frac{r_i}{kr_i - 1} \leq 0$$

For the first inequality we applied $T' < 0$ and $T < \sum_i \frac{1}{kr_i - 1}$. The last inequality follows by applying Lemma 3.1 to the functions $\varphi(r) = \frac{1}{kr-1}$ and $\psi(r) = \frac{r}{kr-1} = \frac{1}{k-1/r}$. □

Hence, we have proved that there is a unique root of G, which we call $\hat{k}$.

4 Newton's method

Newton's approximation method prescribes the iteration

$$k_{n+1} = k_n - \frac{G(k_n)}{G'(k_n)}$$

We will show that G is convex, i.e. $G'' > 0$, and find a point k_0 such that $G(k_0) > 0$. These conditions suffice to guarantee that the method monotonically converges to $\hat{k}$, starting at k_0. If the initial point was chosen between $\hat{k}$ and σ, the next point could result lower than ρ and the method might fail to converge.

Proposition 4.1. *The function G is convex.*

Proof. We retain the notation from the previous proof.

$$G' = T^n P H$$

$$G'' = (T^n P)' H + (T^n P) H'$$

We will show that each term is positive. Recall that $H < 0$.

$$(T^n P)' = T^{n-1} P \left(nT' + T \sum \frac{r_i}{kr_i - 1} \right)$$

We have $T^{n-1} P \geq 0$ and the third factor is negative, as can be seen from the previous proof. For the other term,

$$H = (n+1)T' + T \sum_i \frac{r_i}{kr_i - 1}$$

$$H' = (n+1)T'' + T' \sum_i \frac{r_i}{kr_i - 1} - T \sum_i \frac{r_i^2}{(kr_i - 1)^2}$$

where

$$T'' = 2 \sum_i \frac{r_i^2}{(kr_i - 1)^3}$$

Finally,

$$H' > 2(n+1) \sum_i \frac{r_i^2}{(kr_i - 1)^3} - \sum_i \frac{r_i}{(kr_i - 1)^2} \sum_i \frac{r_i}{kr_i - 1} - \sum_i \frac{1}{kr_i - 1} \sum_i \frac{r_i^2}{(kr_i - 1)^2} > 0$$

For the first inequality, we only use $-T > -\sum_i \frac{1}{kr_i - 1}$. The second inequality follows by applying Lemma 3.1 twice. First use $\varphi_1(r) = \frac{r}{(kr-1)^2} = \varphi(r)\psi(r)$ and $\psi_1(r) = \frac{r}{kr-1} = \psi(r)$. Second, use $\varphi_2 = \varphi$, $\psi_2 = \psi^2$. □

Proposition 4.2. *Let $k_0 = (1 + \frac{1}{1+b})\rho$. We have $G(k_0) > 0$.*

Proof. We need to show $T(k_0)^{n+1} P(k_0) > \alpha$. Recall that $\alpha = ac^{n+1} = bc^n$. Therefore, we want

$$T(k_0) \ (T(k_0)^n P(k_0)) > bc^n$$

We will prove that $T(k_0) > b$ and $T(k_0)^n P(k_0) > c^n$.

We bound the sum in T from below by its largest term.

$$T(k_0) > \frac{1}{(1 + \frac{1}{1+b})\rho \cdot \min r_i - 1} = b$$

Since $\rho = \frac{1}{\min r_i}$.

For the other inequality it suffices to have $T(k_0)(k_0 r_i - 1) > c \ \forall i$.
We use the same bound for $T(k_0)$ and $k_0 r_i - 1 \geq k_0 \min r_i - 1 > 0$. Thus,

$$T(k_0)(k_0 r_i - 1) > \left(\frac{1}{k_0 \min r_i - 1} - 1 \right) (k_0 \min r_i - 1) =$$

$$= 2 - k_0 \min r_i = 1 - \frac{1}{1+b} = \frac{b}{1+b} > \frac{b}{a} = c$$

The last inequality holds since $a = b + n^n$, and $n \geq 2$. □

Acknowledgements: This work was made possible by the entire Balancer team, especially João Bruno and Juan Ignacio Ubeira.

References

[1] Hayden Adams, *Uniswap whitepaper*, https://hackmd.io/C-DvwDSfSxuh-Gd4WKE_ig, 2018.

[2] Guillermo Angeris and Tarun Chitra, *Improved price oracles: Constant function market makers*, arXiv:2003.100001, 2020.

[3] Michael Egorov, *StableSwap - efficient mechanism for Stablecoin liquidity*, (2019).

[4] Felix Leupold, *Development of Manipulation-Resilient Price Oracle for CoW AMM LP Tokens*, (2024).

[5] Fernando Martinelli and Nikolai Mushegian, *Balancer whitepaper*, Technical report (2019).

[6] Jesper Kristensen Miguel Ottina, Peter Johannes Steffensen, *Automated market makers: A practical guide to decentralized exchanges and cryptocurrency trading*, Apress, 2023.