Curve StableSwap Math

9Kynes

February 2022

1 intro

Curve StableSwap is an interesting but somewhat convoluted decentralized exchange. Here, I present a quick and dirt summary of the underlying math.

The key thing to understand about Curve StableSwap is that it exists in between a constant product (Uniswap, x*y=k) and a constant sum (x+y=k). When multiple coins in a Curve pool have close to equal value, curve math is closer to constant sum to decrease slippage. When the coins are farther apart in value, curve math is closer to constant product. This prevents draining one coin from liquidity pools.

2 variables

n: number of **distinct** coins in the pool. Curve supports multi-coin pools.

D: total number of coins in the pool if each were to have an equal price. Thus, $(\frac{D}{n})^n$ represents the standard, Uniswap v2-esque, constant product.

$x_1, x_2, ..$: distinct coins

χ: a variable that determines how much price is determined by constant product versus constant sum. Higher χ means constant sum is weighted higher.

A: a constant that determines overall how close to constant sum the system is. $A = 0$ indicates constant sum is not used at all (the system only uses constant product). The top stablecoin pools have A of 2000, 100, 300, and 100.

3 generalized equations

Stableswap maintains the following equality:

$$\chi D^{n-1} \Sigma x_i + \Pi x_i = \chi D^n + (\frac{D}{n})^n \tag{1}$$

This can be interpreted as follows. χ weights the constant sum (Σx_i). It is multiplied by D^{n-1} to preserve units/dimensions.

Πx_i is the constant product part of the formula.

The right half of the inequality is usually a constant in simpler pools (like constant product or constant sum). Here it varies with χ but is otherwise very similar. Observe that $(\frac{D}{n})^n$ is simply the constant product of all the coins in the pool. χD^n, on the other hand, is equal to $\chi D^{n-1} * D$, which ensures Σx_i stays close to D.

$$\chi = \frac{A\Pi x_i}{(\frac{D}{n})^n} \tag{2}$$

As you can see, χ is A when there is an equal amount of every coin, and decreases towards 0 as the pool becomes unbalanced. A is thus extremely important to determining how much constant sum is used versus constant product.

Putting it all together gives the following equation (directly from the StableSwap white paper):

$$An^n \Sigma x_i + D = ADn^n + \frac{D^{n+1}}{n^n \Pi x_i} \tag{3}$$

4 2-coin solution

The StableSwap white paper asserts that prices are found using an "iterative, converging solution." Using a 2-coin method, we will explore if a more efficient method works.

Let us establish the equation for 2 coins.

$$4A(x_1 + x_2) + D = 4AD + \frac{D^3}{4x_1 x_2} \tag{4}$$

D, in the whitepaper, is a constant, so should be queryable. Or, D can be solved with the invariant equation when one knows token balances and A.

We want to find out how much of x_2 we get back ($-h_2$) if we deposit h_1 of x_1 into the pool.

So we have

$$4A(x_1 + h_1 + x_2 + h_2) + D = 4AD + \frac{D^3}{4(x_1 + h_1)(x_2 + h_2)} \tag{5}$$

We can solve for h_2 with wolfram alpha. The first link is representation of the equation with all constants positive. The second is a solution for h_2.

$link1$

$link2$

When solving for h_2, it is often useful to set another condition of $-h_2 < x_2$ because multiple answers are possible, only one of which is sensible.

The above amount in link 2 should be multiplied by -f with f being a fee (ie. .97) to account for fees. It then gives the amount back as determined by the

equation for putting h_1 in. Note that applying the fee on output is *not* how Uniswap works.

But it is incredibly convoluted so perhaps solving it computationally or iteratively is the way to go.

With sufficiently high A at a price close to 1, a linear approximation may be sufficient.