11 | Automated Market Makers

On June 2017 , Bancor took the crypto-world by storm in selling 79.3 million BNT tokens in a three-hour sale. The product that they were offering was one of crypto's first large decentralized automated exchanges. Up to that point in time, most crypto was traded on (a) hackeable centralized exchanges, or (b) slow and expensive decentralized order-book exchanges. Bancor solved both issues in designing a truly decentralized automatic exchange system.

Automated Market Makers (AMMs) - of which Bancor is a subset - are one of the first dapps that you will get in contact with and as you will use them regularly it's good to have a proper understanding of how they work. AMMs are markets where you can exchange one digital asset (say, an ERC20 on Ethereum) for another, just like one regular exchanges. Unlike traditional exchanges, however, AMMs do not work with an order-book to match supply and demand, but instead clear orders automatically - and almost instantly. The counter-party, in this case, is therefore not a buyer (if you are selling) but rather a smart-contract that you are interacting with. What's more, the smart-contract can take both the role of the buyer as well as that of the seller!

11.1 BASICS

RELAYS

When you swap a token on Bancor, you are interacting with a *relay* smart-contract which acts much like a merchant would on a market, but autonomously.

Let us assume for the sake of example that we are interested in trading bananas for grapes. When the relay sets up shop, it contains equal reserves of both bananas and grapes (by value). For instance, if the *fair price* is 1 banana for 10 grapes, we could initially have 100 bananas and 1000 grapes in the shop. Furthermore, we program the relay to only accept a trade if it swaps exactly 1:10.

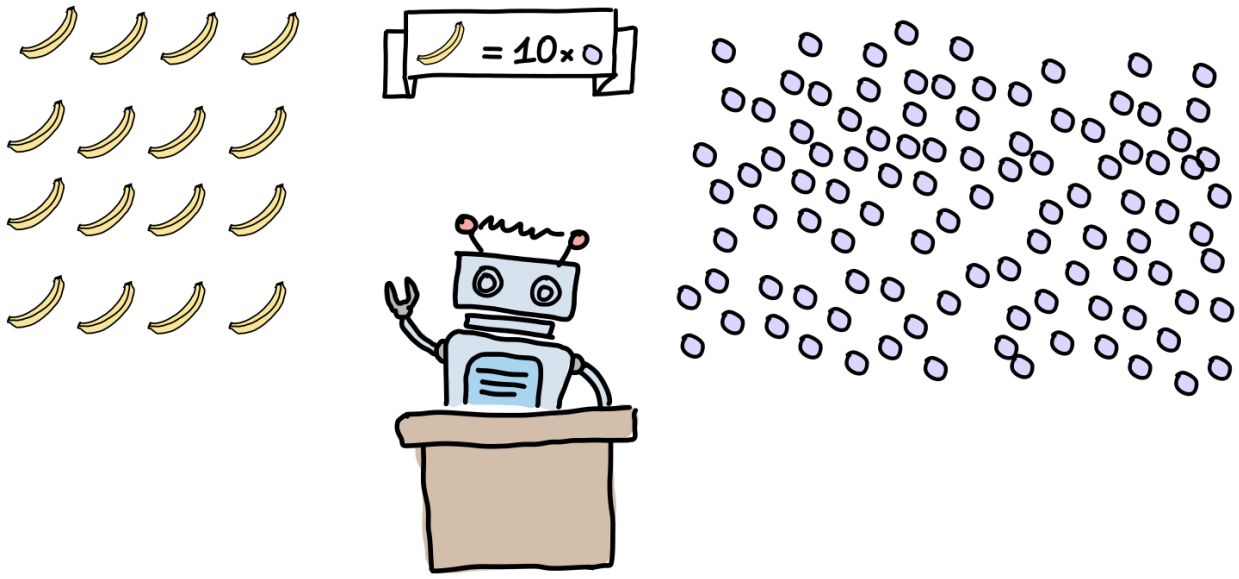
As time goes on, however, the price of grapes and bananas may evolve. Perhaps we had a monkey invasion, and now bananas are a really rare commodity! An important feature of relay contracts is that they dynamically update their price as the supply and demand of the underlying assets change. For instance, after the monkey invasion we may see the price evolve as follows:

t	PRICE (B/G)	#BANANAS	#GRAPES
1	1:10	100	1000
2	1:11	99	1010
3	1:12	98	1021
.....		...	...

At t=1, we still have our original price set. Since bananas are actually much more expensive now, traders quickly try to swap as many bananas as possible on the relay. As the relay's reserves contain increasingly lower amounts of bananas, the relay updates the amount of grapes it wants for 1 banana accordingly.

Relay Network

Bancor was created to provide exchange between a wide variety of assets (397 at the time of writing). Doing so would require 78,606 relays (one for each pair) which is of course not a very efficient way to go about things. Further com-



pounding our problems, there are some assets which may be rare or new. If we fragment them over 396 exchanges, we are bound to run into liquidity issues. In fact, this is the very raison-d'etre for Bancor: to provide liquidity to "long-tail" assets (assets with small volume).

Bancor solves this issue using and invention that was first envisioned by economist Keynes who proposed a supranational currency to facilitate this very same problem in foreign exchange. The idea is simple: create a *common reserve asset* that everybody wants, and that all other assets can trade into (Keynes proposed relating this asset to gold somehow). Let's say, for instance, that China is interested in obtaining dollars, but the U.S. is not interested in obtaining yuan. Instead of dealing directly with the U.S., China could exchange its yuan to the reserve asset and then change the reserve asset for dollars. This way, China received its dollars and the U.S. received the reserve asset which it wants as well. The proposal was widely lauded and subsequently proposed at the infamous Bretton Woods Conference, but failed to gain majority acceptance. In the end, it was decided that all currencies would tie their exchange rates to physical gold and therefore the USD (because USD had the largest gold reserves at the time).

In a similar vein, Bancor ties all currencies to a protocol-specific token called BNT[1]. Moreover, all relays have their own associated reserve currency which can be freely swapped for both BNT and a counter-tokens in the relay. If we want to convert, say, ETH to WBTC we proceed as follows:

1. Exchange ETH for BNT using the ETH relay. We mint ETH relay reserve tokens during this process.

2. Exchange BNT to WBTC using the WBTC relay. We burn BNT relay reserve tokens during this the process.

Note that after each exchange happens, the underlying relays will update their reserves and reserve prices. This process is illustrated below:

In practice, we never interact with the relays ourselves when we perform swaps. Instead, we talk to the Bancor network which takes care of all the details behind the scenes.

[1]Bancor also defines a alternative stable reserve currency called USDB, but we will stick to BNT for the sake of simplicity here.

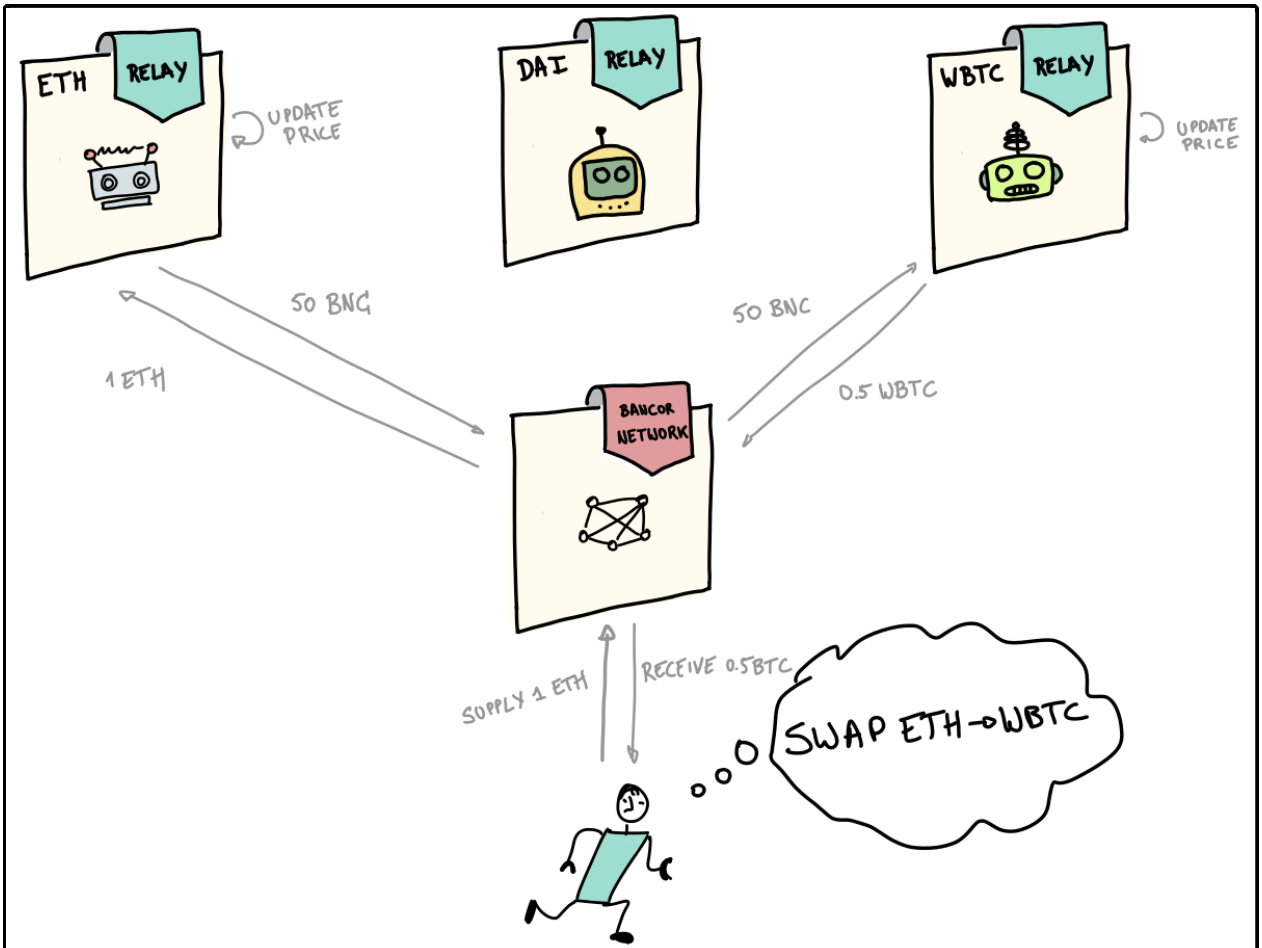


Figure 1: The relay system allows swapping tokens for exotic pairs and avoids the quadratic complexity of needing to have liquidity for everything.

11.2 PRICE MECHANISMS

BONDING CURVES

==TODO: Should we move this to a chapter on ICO's?==

To understand how the prices of our relays are guaranteed to always be a correct reflection of the market price, we must first talk about the relays' reserve currencies and bonding curves.

Monetary Policy of Continuous Supply Tokens

Bonding curves were proposed by Simon de La Rouviere in 2017 as an answer to ICOs. The problem with ICO's is that you mint all tokens at once at a fixed price and a fixed supply, which leads to all sorts of problems including front-running and the inability to issue more tokens in the future for funding purposes. Simon proposed an alternative scheme in which a smart-contract controls the minting, pricing, and burning of the reserve and therefore acts as the main liquidity provider.

The basic idea is that you "bond" (deposit), a

limited amount of existing tokens (say, 10 ETH) into the smart contract, which then mints however many reserve tokens (let's call these RSV) needed to match the initial price that you set for the reserve token. Conversely, you can always swap your RSV back to ETH, which causes the smart contract to (a) reduce its ETH reserve and (b) increase its RSV tokens. These two actions together, in turn, impact the price of RSV. That is, as you add or remove ETH to the contract, the contract updates the price for that ETH based on a curve called the "bonding curve" which defines the *monetary policy* of the reserve token. This is akin to traditional supply / demand curves seen in economics and the previous chapter on stablecoins, but in AMMs it's typical to just look at the resulting bonding curve and liberally experiment with its shape. For instance, we may determine that a relation of $P(S) = (S/10)^2$ is optimal for our reserve token, leading to the following bonding curve:

Continuing our previous example, we see that at the start we provided it 10 ETH at a fixed price of 1 ETH / RSV. Therefore, the starting reserve had a supply of 20 RSV which the creator of the reserve token receiving all of the supply for depositing her 1 ETH. Let's be optimistic and assume that many users believe in the project backed by this reserve token and purchase RSV with their ETH thereby increasing the supply of RSV to 40. If we wanted to now buy RSV for ETH, the price would have increased to 16 ETH / RSV, a 16x increase! This is quite an aggressive pricing schedule, with some notable exceptions people typically tend to stick to linear, or even logarithmic functions.

Pricing a trade

The price (point on the curve) that we have referred to until now is the "instantaneous" price, meaning it's the price if we were to trade an incredibly small amount of ETH/RSV. The *effective price* P_e for trading larger amounts is instead found by somehow "averaging" over the curve from the starting price point to the ending price point as can be seen in Figure 3(left).

For instance, consider that the current price is 16 RSV/ETH, what would the effective price of the trade be if we wanted to buy 10 RSV using ETH? The two quoted prices on the curve are:

$$P_0 = \left(\frac{40}{10}\right)^2 = 16$$
$$P_1 = \left(\frac{50}{10}\right)^2 = 25$$

A crude first-order estimate of the effective price would be to take the average of these two endpoints for our trade: $P_e \approx 20.5$. The correct effective price, is however found by integrating over the price curve:

$$\begin{aligned} P_e &= \frac{1}{50-40}\int_{40}^{50}\left(\frac{S}{10}\right)^2 \mathrm{d}S \\ &= \frac{S^3}{3000}\Bigg|_{40}^{50} \\ &= 41.6 - 21.3 = 20.3 \end{aligned}$$

Note the division by the number of tokens as merely integrating would give us the price for all tokens traded (10 in this case). A more general formula would be:

$$P_e = \frac{1}{S_1 - S_0}\int_{S_0}^{S_1} P(S)\,\mathrm{d}S$$

You will probably never use this formula in your own trading, the important thing to remember, however, is that you pay more/receive less for trades where the amount you trade significantly shifts the price point. This happens when you either (a) trade a lot of tokens, or (b) are trading a token with a small user-base/supply. We will discuss other factors that can impact your trade later in this chapter.

BONDING CURVES IN BANCOR

Smart Tokens

Bonding curves are defined within the relays smart contracts of Bancor which act as *smart tokens*. That is, a smart token represents money

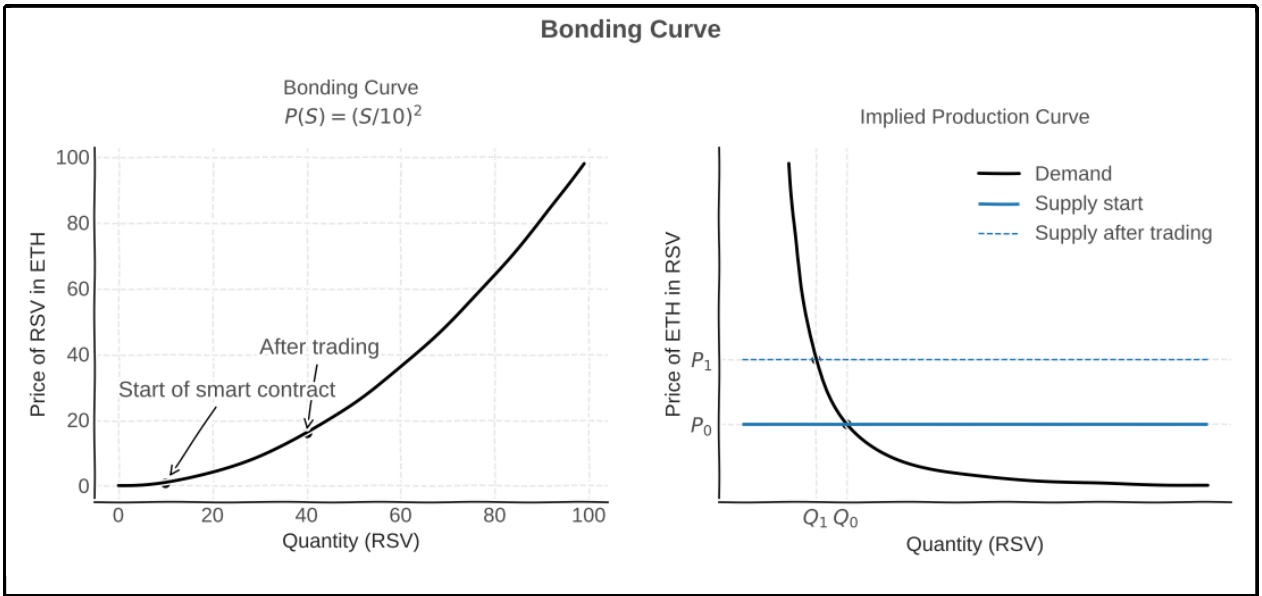


Figure 2: The bonding curve "bonds" the price of the RSV token w.r.t ETH as a function of its supply. More demand implies a higher pprice and vice versa.

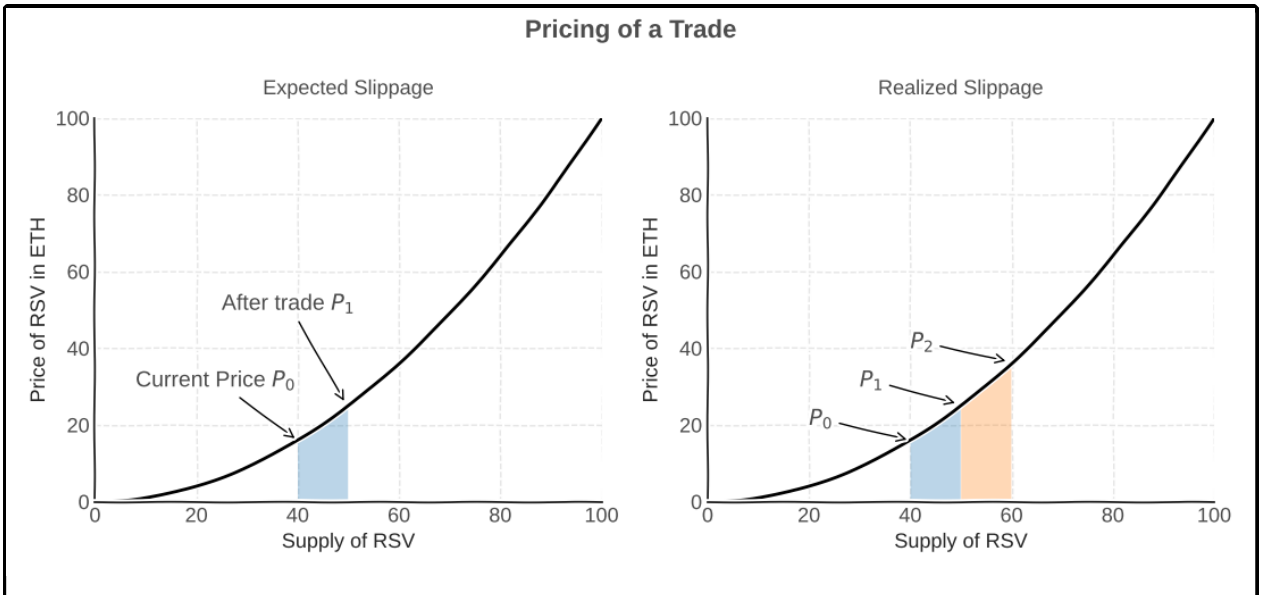


Figure 3: A trade moves the price point along the bonding curve.

(ST) that holds other money in its reserves (let us assume that we use ETH as the "other" for now). If you've studied economy before, you may recognize this scheme as it is similar to how Keynes' supranational currency proposal required gold. Key characteristics of a smart token are[2]:

- Anyone can purchase the smart token using the reserve tokens (ETH). The buyer gets newly minted smart tokens (ST) in return.
- Unlike ICO's and Keynes' model, buyers can liquidate their smart tokens back to the contract in exchange for the original currency at all times.
- A smart token ensures that at all times the *reserve ratio* is fixed (bonded). That is, the ratio of the reserve balance versus the market-cap (supply times price) of the smart token are kept constant.
- The price of the smart token therefore changes with its supply and demand: less demand implies more selling which implies a lower price.

A simple analogy that people sometimes use when discussing this protocol is that you could interpret a deposit of ETH as if the contract "lends" ETH from the user and gives back an I-Owe-You of ST. When the user wants her ETH back, she provides the I-Owe-You and the contract sends back the desired ETH.

Monetary Policy of a Smart Token

As we mentioned above, the monetary policy is such that the ratio of the reserve balance versus the market-cap (supply times price) of the smart token are kept constant. This is encoded as follows in Bancor:

$$F = \frac{R_{\text{ETH}}}{S_{\text{ST}} \times P}$$

[2]https://blog.bancor.network/smart-tokens-101-63edc2cc5a89

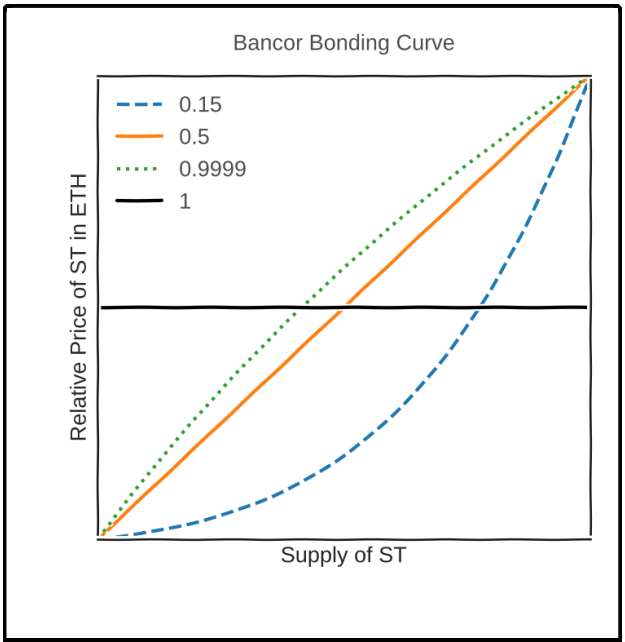


Figure 4: Bancor's bonding curves

Where reserve ratio $F \in [0, 1)$ is a fixed constant chosen at the start, and P the price of the smart-token in ETH. We will drop the subscript for S_{ST} since we only deal with one smart token here. You can show that the bonding curve for such a supply is given by (see appendix for the proof):

$$P(S) = \left(\frac{S}{S_0}\right)^{1/F-1} P_0$$

Where S_0 and P_0 are the initial supply and price respectively. This formula defines a family of bonding curves, specified by the constant reserve ratio F which determines the elasticity of the price (how aggressively the price reacts to changes in supply). Figure 4 shows supply curves for different factors F.

While in most cases F is chosen to be 50% such that we get a linear relation, it's instructive to look at the other cases as well for later:

- At $F \in (0, 0.5)$ we have a decreasingly aggressive price curve. Low values mean that the ST becomes very expensive quickly,

whereas high values mean that the ST becomes more linearly behaved.

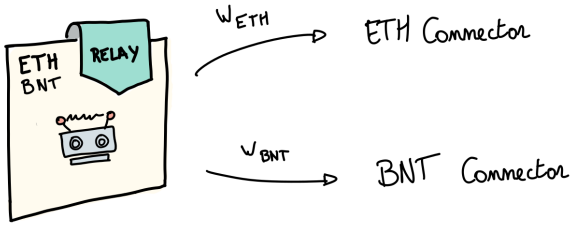
- At $F \in (50\%, 100\%)$ we get a concave price curve that increases its price less than a linear relation would demand.
- $F = 100\%$ is a pathological case where we effectively peg the token to a constant value (i.e., the price never changes, regardless of supply).
- $F = 0\%$ is not a valid choice as it would lead to infinite prices.

Regardless of choice, we can see that by just focussing on the bonding curve, we are already realizing a wide variety of interesting monetary policies.

Multi-Asset Bonding

Bancor adds one more layer of complexity in that it allows for multiple reserves at the same time. We could for instance have a smart token reserve containing both BNT and ETH (just like you could have USD and GBP in Keynes' model). This realization by the Bancor team was the enabler for most of its project: mixing reserves allows us to swap one reserve for the other through the instant minting and liquidation of the underlying smart token currency.

In order to keep things simple, we will isolate just one relay token and its reserves for now (say, a relay that takes care of ETH to BNT conversions). The ETH-BNT relay contains a smart token that connects ETH to BNT through a smart token with a bonding curve as described before.



At the start, these two tokens are added to the reserves according to their respective reserve ratios. Since the sum of all the reserve ratios must be 1 for all of this to work, these are often just set to $F_{\text{ETH}} = F_{\text{BNT}} = 50\%$ in Bancor). A relay, therefore, defines the the bonding curve $P(S)$ for both assets simultaneously as:

$$\begin{cases} P_B(S) = \dfrac{R_{\text{BNT}}}{S \times F_{\text{BNT}}} \text{ BNT / ST} \\ P_E(S) = \dfrac{R_{\text{ETH}}}{S \times F_{\text{ETH}}} \text{ ETH / ST} \end{cases}$$

where $R_{\text{BNT}}, R_{\text{ETH}}$ are the current balances of BNT and ETH in the contract respectively, S the current total supply of the smart tokens reserve. We can easily obtain the value of one ETH in BNT by dividing both quantities:

$$\frac{P_B}{P_E} = \frac{R_{BNT} F_{\text{ETH}}}{R_{ETH} F_{\text{BNT}}} \text{ BNT/ETH}$$

Example: on May 17, 2021, the ETH-BNT relayhad the following values inside of its smart contract[a]:

- ETH reserve: 8086247021951042441764 4
- BNT reserve: 43111821473433462889874866
- ETHBNT supply: 75596117361228700311720358
- $F_{\text{ETH-BNT}} = 50\%$

Therefore, one ETHBNT was worth:

$$P \approx 1.14015 \text{ BNT}$$
$$P \approx 0.002139 \text{ ETH}$$

Which implies that we could get approximately 533 BNT for 1 ETH. This was indeed the case as at the time of writing ETH was priced at 6.4 dollars whereas ETH was priced at 3410 dollars.

[a] Use SmartToken.totalSupply, Converter.reserveBalances()and Converter.reserveWeight()to get these numbers.

For more than two assets T_i, the bonding curves follow a similar pattern. Assuming that the weights sum to one ($\sum_i F_i \leq 1$), the reserve ratios must be:

$$P_i = \frac{R_i}{P_i \times F_i}$$

Pairwise prices can be derived by dividing the currencies using the same logic as before.

Swapping dynamics

When we buy a token from the relay (e.g., we buy ETH for BNT), we reduce the reserve of that token in the smart contract R_{ETH}, as well as the supply of the Smart Token S. The counter-reserve of R_{BNT} remains unchanged. According to our monetary policy, ETH has then become more expensive relative to BNT (which has become inversely proportional more expensive to the increase in supply). This is best understood using an example.

Consider for instance, that we start with a supply of $S = 2000$ ST and reserves containing $R_{\text{ETH}} = 100$ and $R_{\text{BNT}} = 1000$ and assume equal reserve ratios of $F = 50\%$. The initial price of the reserve currency is then:

$$\begin{cases} P_B(S) = \dfrac{1000}{2000 \times 50\%} = 1 \text{ BNT / ST} \\ P_E(S) = \dfrac{100}{2000 \times 50\%} = 0.1 \text{ ETH / ST} \end{cases}$$
$$\implies \frac{P_B}{P_E} = 10 \text{ BNT/ETH}$$

If we were to buy 1 ETH from the contract we reduce the reserve currency as well as the smart token supply to obtain:

$$\begin{cases} P_B(S) = \dfrac{1000}{(2000-1) \times 50\%} = 1.0005 \text{ BNT / ST} \\ P_E(S) = \dfrac{(100-1)}{(2000-1) \times 50\%} = 0.099 \text{ ETH / ST} \end{cases}$$
$$\implies \frac{P_B}{P_E} = 10.1 \text{ BNT/ETH}$$

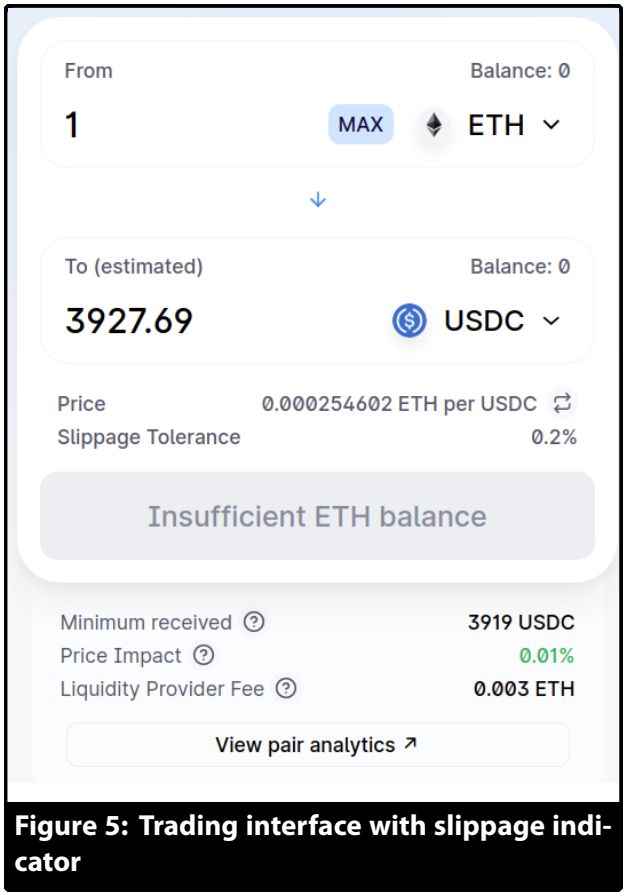


Figure 5: Trading interface with slippage indicator

Otherwise put, the action of buying 1 ETH from the smart contract indicates to it that ETH is now more in demand. Subsequently, the smart-contract has rebalanced its price slightly upward.

11.3 BOTS, SLIPPAGE, AND FEES

When you are initiating a trade on a protocol like Bancor, Uniswap, Sushiswap, Quickswap, etc, you will be presented with a screen like the one below:

We already understand how the basic quotation of a swap is calculated, but as you can see it seems like this price is not necessarily guaranteed as we have field for "Minimum received" which is 8 dollars lower than the quoted price! Similarly, if you were to reverse this pair (buy ETH for USDC), you would see quite a spread on the quoted price. All this can seem a bit like magic, or perhaps even manipulation when you

start out so it's instructive to take a look at why these differences exist.

Slippage is the difference between the expected price of a trade and the price at which the trade is executed. When you swap tokens on blockchain dapps as shown in the above screenshot, you will typically be presented with a slippage tolerance level (0.2% in the picture, but the default is typically around 0.5%). The slippage tolerance tells the smart contract that you are willing to accept trades that go up to 0.2% lower than what is being quoted in the interface. The good news is that you can set it to whichever number you want, the bad news is that if you set it too low, your trade might not be executed as the smart contract will work to ensure that you transaction either executes as specified, or fails with error code: "*INSUFFICIENT_OUTPUT_AMOUNT*" or similar.

There are 4 sources of deviation from the theoretical price:

1. Trading fees paid to the liquidity providers (fixed % loss).
2. Curve slippage
3. Volatility and liquidity of the currency being traded.
4. Liquidity of the pool (set slippage higher in times).
5. Arbitrageurs trying to front-run trades.

LIQUIDITY PROVIDER FEES

We will talk extensively about how to put your own capital into AMMs later, but what's important for us here, is that there are people who put their crypto into the smart contracts' reserves (*liquidity providers*). Naturally, they want to see some return for the costs involved in parking their capital (including *opportunity costs* and smart contract risks). Trading fees range from 0.2-0.6% per trade on most platforms. The liquidity provider fee is usually already priced-in into the quoted price.

CURVE SLIPPAGE

As we've discussed before, trading large amounts on the bonding curve results in a potentially significant change in reserves which we need to average out over to obtain the effective price. An easy approximation here is to average out the before-and-after situation, or you can integrate the result over the bonding curve like we did before. For Bancor, the effective price of a trade is given by:

$$P = \frac{\Delta A_{\text{out}}}{\Delta S}$$

$$\Delta A_{\text{out}} = S\left[\left(1 + \frac{A_{\text{in}}}{R_{\text{in}}}\right)^{F_{\text{in}}} - 1\right]$$

$$\Delta S = R\left[\sqrt[F]{\left(1 + \frac{S_{\text{destroyed}}}{S}\right)} - 1\right]$$

Where A_{in}, A_{out} are the amount of input and output tokens of the trade, and $S_{\text{destroyed}}$ the amount of smart tokens burned in the process. For a derivation of this result, see Rosenfeld 2017. The smaller the liquidity of the pool that you are trading in, the larger the effect of this source of slippage is. If you are trading in very small pools, you are likely making substantial dents in the reserves for every trade. Therefore, you may have to split-up your orders and wait a long time for reserves to go back to levels in between your orders to obtain an effective price close to your initially quoted price.

VOLATILITY & FRONT-RUNNING

Just like on traditional markets, you will likely face less reliable price quotations when you are trading during a period of high volatility (e.g., after an important news came out, or during a bull run). The reason is that the order of execution of trades is determined by how much gas you pay for your order. It is possible that someone else has paid more gas than you to execute the same order, and therefore was placed first in line. This could happen by accident, or on purpose by bots

that are trying to arbitrage your trade [3]. Your order will then be quoted after that other trade and likely be more expensive. The lower the average gas cost at the time of transaction, the higher the impact of this effect (as there is likely more bot-activity).

Figure 3 right shows the effect of front-running on your trade. The transaction that happened before you realized a price of P_1 but your transaction - which happened afterwards - was priced at P_2.

It is advisable that you avoid trading at such moments, especially in low-liquidity pools. If you must, you can:

1. Split up your transaction into smaller trades to limit the amount of slippage due to volatility of the order.

2. Set the slippage tolerance-level higher to ensure that your order can pass.

3. Set the gas cost high enough so that the order can pass through faster, but bots can monitor your gas bid and still try to outbid you.

Of course, every transaction costs money as well. As you trade more, you will gain intuition for how to balance the above three to figure out the most economical way to transact (though you shouldn't shy away from a spreadsheet when in doubt!).

11.4 OVERVIEW OF POPULAR AMMS

Once you understand Bancor, it's relatively easy to understand all the other AMMs which evolved afterwards. The two most important alternatives to know in the space are Uniswap and Curve. Uniswap is by far the most popular and currently has 10x the volume of Bancor. It evolved as an answer to the criticism that Bancor is inefficient, and slow: exchanging through a couple of relays using an intermediary currency is gas-intensive, and the white-listing / security check for tokens is sluggish for a space that moves as quickly as DeFi.

UNISWAP

The constant-product rule (invariant)

The basic premise behind Uniswap is the *constant-product-rule* which states that all points in time, the reserves for two currencies (say, R_x and R_y) in a pool should be equal to a constant k that is defined at the creation date of the pool:

$$k = R_x \times R_y$$

The reserves can go up and down over time as trades happen to the exchange, but no matter what happens k is kept constant at all times. This is, of course, a special case of Bancor's bonding curve. The implied price of x's to y's for small trades is:

$$P = \frac{R_x}{R_y}$$

> **Example**: At the time of writing the total Uniswap USDC-ETH pool contained $R_y = 54,936,643$ million USDC and $R_x = 14,556$ ETH . Therefore, the price of ETH at that point in time was:
>
> $$P = \frac{54,936,643 \text{ USDC}}{14,556 \text{ ETH}} \approx 3774 \text{ USDC/ETH}$$

Invariant Reserves

The constant product formula can also be reversed to figure out what the reserve price would be for an asset when the price P hits a certain level. Assuming that there are no pathological scenarios like liquidity constraints, the reserve values can be written in terms of the current

[3] This is called *priority-gas auctioning*, and it's crypto's equivalent of high-frequency flash trading. It's usually executed by miners (Miner-Extractable Value) and is actively being researched, see the following paper for an excellent review: https://arxiv.org/pdf/1904.05234.pdf.

price P:

$$\begin{aligned} R_x &= \frac{k}{R_y} \\ &= \frac{k}{P \times R_x} \\ &= \sqrt{\frac{k}{P}} \end{aligned}$$

Similarly:

$$R_y = \sqrt{k \times P}$$

> **Example:** we're trading ETH and USDC. the pool started at a balance of 1,000 USD/ETH and contains 5,000 USDC and 50 ETH:
>
> $$\begin{aligned} R_x &= 50 \\ R_y &= 50,000 \\ k &= R_x \times R_y = 2,500,000 \end{aligned}$$
>
> Over time, the price doubled to 2,000 USDC/ETH, therefore, the reserves are now:
>
> $$\begin{aligned} R_x &= \sqrt{2,500,000/2,000} \approx 35 R_y \\ &= \sqrt{2,500,000 \times 2,000} \approx 7071 \end{aligned}$$
>
> We can confirm that k is indeed still 2,500,000 as required by the protocol.

While all other dynamics are very similar to bancor, it's useful to talk about the implied *invariant curve* of the above rule. The constant multiplicative rule leads to the reserve schedule shown in Figure 6. At all points in time, the reserves must lie on the blue line. That is, they must lie on the (reserve) *invariant curve*.

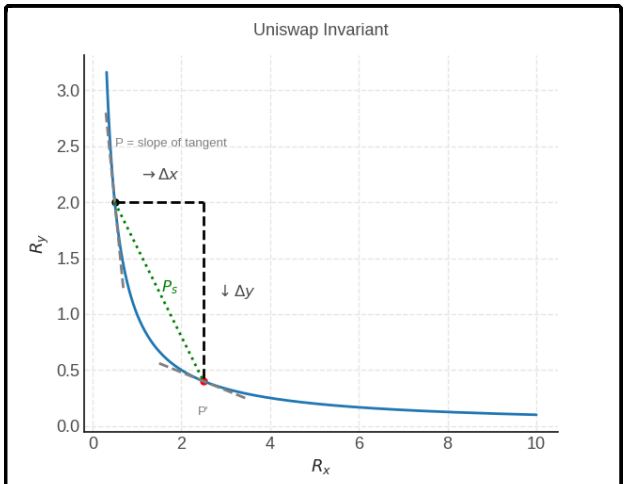


Figure 6: Uniswap's invariant curve

Swapping dynamics using the invariant

If someone wants to swap x's to y's, that person will cause a shift in the reserves (adding, say, Δx to R_x and removing Δy from R_y). This shift leads to an *exchange rate* for the swap that is equal to:

$$\begin{aligned} P_s &= \frac{R'_y - R_y}{R'_x - R_x} \\ &= \frac{R_y + \Delta y - R_y}{R_x + \Delta x - R_x} \\ &= \frac{\Delta y}{\Delta x} \end{aligned}$$

We will calculate Δy in the next section, but for now observe that this ratio is simply the slope of the line that crosses the reserves before and after the trade (indicated by a green line in the above plot).

The *implied price* P at a certain reserve level is then determined as the "spot" exchange rate, or, the rate that you would get by trading a really small amount Δx (so small that it barely impacts the reserves). This is, of course, just the derivative of the curve itself:

$$\begin{aligned} \lim_{\Delta x \to 0} P_s &= \frac{\mathrm{d}y}{\mathrm{d}x} \\ &= \frac{\mathrm{d}}{\mathrm{d}x} \frac{k}{x} \\ &= \frac{k}{x^2} \\ &= \frac{y}{x} \end{aligned}$$

Buy and sell orders on Uniswap

There are two ways to specify Uniswap orders: (a) you supply an exact amount of input token Δx, and want as much Δy as possible in return or (b) you supply as little as Δx possible to receive exactly The counter-amount received is calculated at the price ratio *post-swap*, but that differs depending on the case.

Case a: supplying Δx to receive as much Δy as possible

We supply exactly Δx, therefore, the reserves will increase to $R'_x Rr_x + \Delta x$ The amount you will receive is:

$$\begin{aligned}(R_x R_y) &= (R_x + \Delta x)(R_y - \Delta y) \\ &\Downarrow \\ \Delta y &= \Delta x \times \frac{R_y}{R_x + \Delta x}\end{aligned}$$

Note that in reality on top of that, a trading fee is added such that:

$$\begin{aligned}(R_x R_y) &= (R_x + \Delta x \cdot (1-f))(R_y - \Delta y) \\ &\Downarrow \\ \Delta y &= (1-f) \cdot \Delta x \times \frac{R_y}{R_x + (1-f) \cdot \Delta x}\end{aligned}$$

> Example: assume no trading fees, a pool with $R_x = 50$ ETH and $ R_y=5,000 ,USDC$, you swap $s_x = 2$ ETH and receive:
>
> $$s_y = \frac{2 \times 5000}{50 + 2} = 192.31 \text{ USDC}$$
>
> This is lower than the original ratio (200 USD/ETH vs 192.31 USD/ETH)

Case b: supplying as little Δx as possible to receive exactly Δy

To receive exactly Δy:

$$\Delta x = \frac{\Delta y \times R_x}{R_y - \Delta y} + 1$$

TODO: explain this formula better

SUSHISWAP, PANCAKESWAP, QUICKSWAP, ...

SushiSwap has been described as a scam, plagiarism, and even the world's first legal billion dollar heist. At the time of its creation, Uniswap was by far the largest exchange, but the community was upset by the way Uniswap was distributing its value with most of it going to VC funds and early investors as opposed to the users themselves. Thus, someone by the name of "Chef Nomi" decided to fork (copy) Uniswap, but make it more community-owned. Original liquidity was stolen from Uniswap through a two-step *vampire-attack*: (1) convince Uniswap liquidity providers to stake their *liquidity tokens* (Uniswap's smart tokens) into Sushiswap (2) convert the liquidity tokens into Sushiswap-native ones. After a turbulent and controversial history, it now stands shoulder-to-shoulder to Uniswap in terms of liquidity.

Other forks have popped-up and will likely continue to pop-up, including Quickswap (on the Polygon L2), PancakeSwap (on BNB), etc.

GENERALIZATION

The Uniswap invariant can be generalized to token pools with more than asset as is done by the Balancer protocol. In this case, the invariant becomes:

$$k = \prod R_i^{w_i}$$

STABLESWAPS

Stableswap protocols like Curve specialize in stablecoins. If we were to blindly apply the constant-product invariant to a reserve with a total starting value of R stablecoins and equal assets of every one of the N stablecoin, our product-invariant would be:

$$\left(\frac{R}{N}\right)^N = \prod_{\text{i=1}}^{N} R_i$$

While the model presented thus far works very well for most tokens, the slippage that it intro-

duces turned out to be prohibitively expensive for stablecoins. That is, swapping large amounts of stablecoins would often incur a price deviation that is not justifiable given their stability. It is intuitively clear that something close to 1 DAI = 1 USDC = 1 USDT, etc should make more sense. A more rigid expression could be a *constant-sum invariant*, e.g.:

$$k = \sum R_i$$

But, this would simply set the price to 1 for every coin as $P = -\frac{\mathrm{d}\, R_i}{\mathrm{d} R_j} = 1$, regardless of the current reserve balances and chosen coins. This is too rigid and would not properly adjust for other risks (like Tether crashing, or someone emptying the reserves to arbitrage another protocol). The invariant that curve therefore proposes is a mix between the constant-product invariant and a constant-sum invariant:

$$f \sum_{i=1}^{N} x_i + \prod_{i=1}^{N} x_i = C$$

Where f is a constant set by the protocol that determines how much importance is given to constant price stability. Thus, we obtaine a spectrum from constant pricing ($f = \infty$) to multiplicative pricing ($m = 0$) (verify!). The above equation can be rewritten to:

$$AN^N \sum R_i + R = ARN^N + \frac{R^{N+1}}{N^N \prod R_i}$$

where A is a constant related to f (for a derivation see the appendix, in the inital release, A was set to 85 for pure stablecoin pools). The implied curves are shown in Figure [4]. Note how it mimicks the constant price curve in the beginning but becomes less forgiving as we move further way along the curve. Just like Bancor's bonding curve, this curve's rigidity can be customized using parameter A.

[4]Note that the curve.fi logo is just a Klein bottle and not related to this formula.

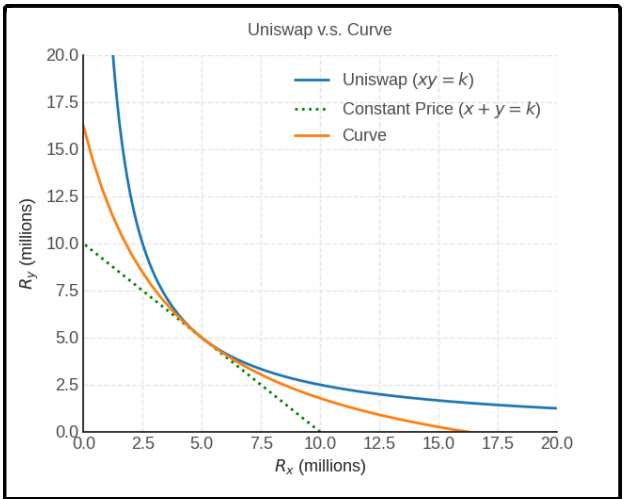


Figure 7: Curve's stableswap invariant lies closer to the 1-to-1 ratio and therefore induces less slippage.

UNISWAP V3.0

T.B.A.

AMM AGGREGATORS

While all of these AMMs are supposed to discover the real price eventually due to arbitrage, it may be that there is some divergence at times. Therefore, when you are doing your trades you may want to check aggregators which aggregate liquidity from various platforms. Two types exist:

- Off-chain aggregators (1inch, paraswap) are flexible and can often trade accross chains. They require more trust from the user as the centralization of the server router can lead to hard-to-detect front-running.
- On-chain aggregators (swapswap) combine other liquidity providers fully on-chain using smart contract logic. They are transparent and unlikely to lead to front-running, but are not as efficient in terms of gas.

1inch, paraswap

11.5 APPENDIX

BANCOR PRICE DERIVATION

The original derivation is defined in Rosenfeld 2017, but we repeat it here with our notation. We consider a smart token ST with a total supply of S and underlying assets R of counter-currency ETH. By definition, we know that these are related as follows:

$$P = \frac{R}{S \times F}$$

We are interested in knowing how an infinitesimal purchase of $\mathrm{d}S$ tokens impacts the supply and price (negative values for $\mathrm{d}S$ would represent a sale). The user will pay $P\,\mathrm{d}S$ for these tokens and shift the reserve by $\mathrm{d}R = P\,\mathrm{d}S$. Let us combine us with the above observation that R = PSF and therefore $\mathrm{d}R = \mathrm{d}PSF$ to obtain:

$$\begin{aligned} P\,\mathrm{d}S &= \mathrm{d}PSF \\ &= F(P\,\mathrm{d}S + S\,\mathrm{d}P) \end{aligned}$$

In the second step we observed that F is a constant and then applied the chain rule of derivatives. Collect all $\mathrm{d}S$ on the left-hand side to obtain:

$$\begin{aligned} P\,\mathrm{d}S(1-F) &= FS\,\mathrm{d}P \\ P\,\mathrm{d}S\left(\frac{1}{F} - 1\right) &= S\,\mathrm{d}P \end{aligned}$$

We are getting close to our formula, let's replace $\alpha = \left(\frac{1}{F} - 1\right)$ and work towards integration:

$$\begin{aligned} \alpha P\,\mathrm{d}S &= FS\,\mathrm{d}P \\ \alpha\frac{\mathrm{d}S}{S} &= F\frac{\mathrm{d}P}{P} \\ \alpha\,\mathrm{d}\log S &= F\,\mathrm{d}\log P \end{aligned}$$

Where the last step applied the logarithmic rule of derivatives ($\mathrm{d}\log x = 1/x$). Integrating both sides we obtain:

$$\begin{aligned} \int \alpha\,\mathrm{d}\log S &= \int \mathrm{d}\log P \\ \alpha\log S + C &= \log P \end{aligned}$$

for some constant C. Exponentiating, we obtain that:

$$\begin{aligned} e^C S^\alpha &= P \\ P &= \left(\frac{S}{S_0}\right)^\alpha P_0 \end{aligned}$$

In the last step, we decomposed constant e^C into an initial price-to-supply ratio P_0/S_0^α. The beauty of this latter formula is that it allows us to avoid difficult integration which would be very costly to perform on the blockchain.

THE CURVE INVARIANT FUNCTION

Curve combine the constant-product and the constant-sum invariants.

$$f\sum_{i=0}^{N} x_i + \prod_{i=0}^{N} x_i = C$$

Where f is a mixing factor that determines how much linear we want to make our function. We always start with a uniform supply of all tokens. The constant is determined with this supply and therefore:

$$f\sum_{i=0}^{N} x_i = R$$
$$\prod_{i=0}^{N} x_i = \left(\frac{R}{N}\right)^N$$

Our initial formula is therefore:

$$f\sum_{i=0}^{N} x_i + \prod_{i=0}^{N} x_i = fR + \left(\frac{R}{N}\right)^N$$

This mix is awkward because the mixing parameter is not dimensionless, meaning, it will function differently depending on how the reserves work out (the sum-terms are linear, whereas the multiplicative ones are powers). In order for interpretability and consistency, the smart-contracts and the paper of Curve use an "amplification factor" instead, which is derived from f.

$$f = \frac{A\prod x_i}{(R/N)^N}$$

Substitution into the formule yields:

$$\left(R\frac{A\prod x_i}{(R/N)^N}\right)\sum_{i=0}^{N} x_i + \prod_{i=0}^{N} x_i = \left(R\frac{A\prod x_i}{(Q/N)^N}\right)R + \left(\frac{R}{N}\right)^N$$

Which can be simplified to:

$$N^N A(\sum x_i) + R = N^N AR + \frac{R^{N+1}}{N^N \prod x}$$

The final equation then becomes:

$$AN^N\sum x_i + R = ARN^N + \frac{R}{N^N \prod x_i}$$

11.6 EXERCISES

Exercise: go back to the bonding curve for bitcoin, estimate its F-value.

11.7 RESOURCES

- https://miguelmota.com/blog/understanding-stableswap-curve/