GitHub: chanhosuh/notes-on-crypto
commit `2a178c0` on branch `master`.

Notes on AMMs

Chan-Ho Suh (*chanho@email.com, chanhosuh.com*)

March 28, 2021

These notes are intended to give a concise introduction to "automated market-makers" (AMMs) in the cryptocurrency industry. It is aimed at those with some experience with cryptocurrency and a modicum of mathematical training, e.g. smart contract developers trying to understand the theory and mathematics behind the protocols.

A brief history

Prediction markets such as Gnosis and Augur ran into the problem of how to set the initial liquidity for their bets. Each token represents an outcome for a particular bet and the relative pricing of outcomes must be set to provide sufficient liquidity for subsequent betting.

Multiple solutions, notably LMSR, were proposed for this problem, but the more general problem of providing decentralized liquidity gained wider attention when Vitalik Buterin made several posts, starting in 2017, on decentralized exchanges.

The key insight he provided was that there was utility in having a simple on-chain market-maker that only required transactions for the actual trades, not for order placement or orderbook management. A mathematical formula would maintain the orderbook state on-chain.

A frequent objection one finds on the early discussion threads is that an AMM would be too inefficient and costly for actual trading. One commenter remarks that it would be too difficult to mitigate frontrunning by miners. The potential for composability of AMMs with other building blocks was not yet understood.

Vitalik responded that a simple implementation that allowed smart contracts to engage instantaneously in swapping tokens would have tangible benefits to some users:

> "The point is not for this kind of exchange to be the only exchange; the point is for it to be one type among many. It offers the benefits of executing a complete trade in one transaction, and extreme user-friendliness even to smart contracts, which are very real benefits and will at least sometimes exceed the costs of slippage to some users."

https://ethresear.ch/t/improving-front-running-resistance-of-x-y-k-market-makers/1281/4

The earliest proposals, such as Vitalik's, had somewhat complex formulae, but Martin Koppelmann (Gnosis) suggested to Vitalik that a constant product formula was simple and had the desired properties for market-making:

- there is *always* liquidity for either token

- the resulting price should reflect the demand
- market-making fees are easily incorporated

With regards to the second item (market impact), with the constant product formula, the ratio of token reserves gives the price of one token vs another (for an infinitesimal quantity). This has the benefit of simplicity but results in a fairly large amount of slippage, as compared to traditional markets and market-making, e.g. in equities.

An interesting property of this type of market-maker is that it is path-independent, i.e. no matter what sequence of trades ensues to drain X amount of token A out and Y amount of token B out, the resulting reserves will be the same. This protects the liquidity-providers from large bubble-like rises and crashes, which ordinarily would be handled in a traditional, centralized market by "running away" (pulling liquidity).

It is interesting to note that the motivation for Vitalik's post was to describe frontrunning mitigation by miners and the usage of "virtual" token reserves to do so. However, what in fact happened was that the post caught interest of Haayden Adams, who then created Uniswap based on the idea, with no frontrunning mitigation in place.

https://ethresear.ch/t/improving-front-running-resistance-of-x-y-k-market-makers/1281

The Uniswap protocol

Uniswap is the original constant-product AMM which originated out of Vitalik's influential AMM post. Vitalik gave much input into its design and implementation (which explains why the original V1 platform is written in Vyper!).

In Uniswap, each *liquidity pool* consists of two tokens, each held in some quantity. When a swap is made, one token is added in some quantity, increasing its reserve, and another is taken out in some quantity, decreasing its reserve. The quantity that is taken out depends on the quantity being put in and the size of the two reserves. It is given by the formula:

$$y_{out} = y \cdot x_{in\_fee} / (x + x_{in\_fee}) \tag{1}$$

x and y	represent the two token reserve amounts.
x_{in}	amount transferred in
x_{in_fee}	x_{in} with 30 basis point fee deducted
y_{out}	amount transferred out

The above formula is the one used in practice for swapping, but it is helpful to know it is equivalent to:

$$(x + x_{in\_fee})(y - y_{out}) = x \cdot y \tag{2}$$

This equation shows the product of the two reserve quantities is maintained before and after the swap (ignoring fees). The product is usually denoted k and is called the Uniswap invariant.

Thus we have the famous constant-product formula:

$$x \cdot y = k \tag{3}$$

In reality, since x_{in_fee} has the fee taken out, the product of the post-swap reserves, $(x + x_{in})(y - y_{out})$, is greater than k. So the "invariant" increases after each swap due to fees. k will also increase or decrease whenever liquidity is added or removed, respectively.

When adding liquidity to the pool, it is assumed the proportion of tokens being added is the same as the proportion of the reserves, i.e. if $x/y = \alpha$, then $\Delta x/\Delta y = \alpha$, where Δx and Δy are the amounts being added. Adding liquidity *mints* a liquidity provider (LP) token, which represents the share of the pool (and its fees) the LP is entitled to. The amount of LP tokens minted, l, is given by:

$$l = \left(\frac{\Delta x}{x}\right) L \tag{4}$$

where:

x	is the token reserve for token X (pre-deposit)
L	is the total supply of LP tokens (pre-deposit)

https://vitalik.ca/general/2017/06/22/marketmakers.html
https://github.com/Uniswap/uniswap-v1/blob/master/contracts/uniswap_exchange.vy

The Balancer protocol

The Curve protocol