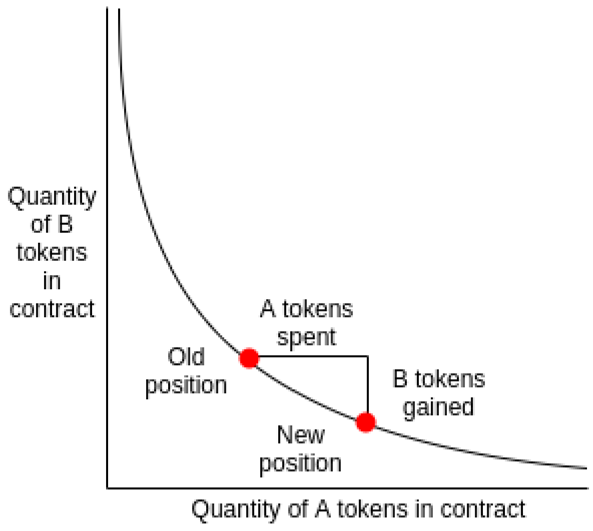
Improving front running resistance of x*y=k market makers

vbuterin #1 March 2, 2018, 9:24am

Two years ago I **made a post** where I suggested that we could run on-chain decentralized exchanges in a similar way to what Augur and Gnosis were proposing for on-chain market makers (eg. LMSR), and Martin Köppelmann from Gnosis suggested a simple approach for doing so, that I call the "x*y=k market maker". The idea is that you have a contract that holds x coins of token A and y coins of token B, and always maintains the invariant that x*y=k for some constant k. Anyone can buy or sell coins by essentially shifting the market maker's position on the x*y=k curve; if they shift the point to the right, then the amount by which they move it right is the amount of token A they have to put in, and the amount by which they shift the point down corresponds to how much of token B they get out.


Quantity of B tokens in contract
Old position
A tokens spent
B tokens gained
New position
Quantity of A tokens in contract

Notice that, like a regular market, the more you buy the higher the marginal exchange rate that you have to pay for each additional unit (think of the slope of the curve at any particular point as being the marginal exchange rate). The nice thing about this kind of design is that it is provably resistant to money pumping; no matter how many people make what kind of trade, the state of the market cannot get off the curve. We can make the market maker profitable by simply charging a fee, eg. 0.3%.

However, there is a flaw in this design: it is vulnerable to front running attacks. Suppose that the state of the market is (10, 10), and I send an order to spend one unit of A on B. Normally, that would change the state of the market to (11, 9.090909), and I would be required to pay one A coin and get 0.909091 B coins in exchange. However, a malicious miner can "wrap" my order with two of their own orders, and get the following result:

- Starting state: (10, 10)
- Miner spends one unit of A: (11, 9.090909), gets 0.909091 units of B
- I spend one unit of A: (12, 8.333333); I get 0.757576 units of B
- Miner spends 0.757576 units of B: (11, 9.090909), gets 1 unit of A

The miner earns 0.151515 coins of profit, with zero risk, all of which comes out of my pocket.

Now, how do we prevent this? One proposal is as follows. As part of the market state, we maintain two sets of “virtual quantities”: the A-side (x, y) and the B-side (x, y). Trades of B for A affect the A-side values only and trades of A for B affect the B-side values only.

Hence, the above scenario now plays out as follows:

- Starting state: ((10, 10), (10, 10))
- Miner spends one unit of A: ((11, 9.090909), (10, 10)), gets 0.909091 units of B
- I spend one unit of A: ((12, 8.333333), (10, 10)); I get 0.757576 units of B
- Miner spends 1.111111 units of B: ((12, 8.333333), (9, 11.111111)), gets 1 unit of A

You still lose 0.151515 coins, but the miner now *loses* 1.111111 - 0.909091 = 0.202020 coins; if the purchases were both infinitesimal in size, this would be a 1:1 griefing attack, though the larger the purchase and the attack get the more unfavorable it is to the miner.

The simplest approach is to reset the virtual quantities after every block; that is, at the start of every block, set both virtual quantities to equal the new actual quantities. In this case, the miner could try to sell back the coins in a transaction in the next block instead of selling them in the same block, thereby recovering the original attack, but they would face competition from every other actor in the system trying to do the same thing; the equilibrium is for everyone to pay high transaction fees to try to get in first, with the end result that the attacking miner ends up losing coins on net, and all proceeds go to the miner of the next block.

In an environment where there is no sophisticated market of counter-attackers, we could make the attack even harder by making the reset period longer than one block. One could create a design that’s robust in a wide variety of circumstances by maintaining a long-running average of how much total activity there is (ie. sum of absolute values of all changes to x per block), and allowing the virtual quantities to converge toward the real quantity at that rate; this way, the mechanism is costly to attack as long as arbitrageurs check the contract at least roughly as frequently as other users.

A more advanced suggestion would be as follows. If the market maker seems to earn profits from the implied spread from the difference between the virtual quantities, these profits could be allocated after the fact to users who seem to have bought at unfair prices. For example, if the price over some period goes from P1 to P2, but at times in between either exceeds P2 or goes below P1, then anyone who bought at that price would be able to send another transaction after the fact to claim some additional funds, to the extent that the market maker has funds available. This would make griefing even less effective, and may also resolve the issue that makes this kind of market maker fare poorly in handling purchases that are large relative to its liquidity pool.

14 Likes

Limit orders and slippage resistance in x*y=k market makers

Update on the USM "minimalist stablecoin": two new features

kladkogex #2 March 2, 2018, 1:18pm

vbuterin:

> One could create a design that's robust in a wide variety of circumstances by maintaining a long-running average of how much total activity there is (ie. sum of absolute values of all changes to x per block), a

Actually when you talk about doing long-running averages, your market maker starts to look like the market maker described **on this message thread** some time ago.

This market maker collects deposits and trades at time 0:00. It is provably fair (everyone eventually gets the market exchange rate) and provably secure against front-running. In fact all participants during a particular day get the same exchange rate.

I really think that these types of on-chain exchanges are superior to "reserve-curve-based" exchanges and will end up being widely used

Here is a short description:

1. Intraday all market participants deposit funds and submit orders.
2. At time 0:00 an exchange happens , where the exchange rate is determined by a balance of deposits
3. Intraday (during a deposit) the party can get an intra-day loan from the market maker in the amount of 80% of the estimated exchange value as determined by the previous day exchange rate. The loan is repaid when the exchange happens at time 0:00
4. The market maker charges fees which increase if the market maker starts losing reserves. The market maker is mathematically guaranteed to never run out of reserves (the fees increase to infinity when the market maker starts losing reserves(
5. Due to arbirtrage and since the deposits are public, at time 0:00 the exchange rate will very much match external exchanges.

Here is a prototype implementation (it is opensource under GPL 2.0 license)

Fair market maker prototype implementation

1 Like

MicahZoltu #3 March 2, 2018, 1:27pm

That type of exchange (where all of one asset is traded for all of the other asset at the end of the day) needs a mechanism for limit orders. As a user, I may be interested in liquidating a large amount of asset ABC. I want to be able to put it all on the exchange but specify the exchange rate I'm willing to accept. When it comes time to settle the days trades, only as much of my asset that can be traded while still getting my target rate will be considered eligible for exchange, the rest will rollover to the next day.

Of course, doing that on-chain is prohibitively expensive, and we don't want to require the user to leg in throughout the day as people fill the other side because that is prohibitively expensive for the participant (gas) and complex (requires many signed transactions).-

2 Likes

vbuterin #4 March 2, 2018, 1:59pm

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. I am ok with just accepting that this kind of approach will not be acceptable to whales who want to liquidate large amounts of things; that’s not the target market.

2 Likes

kladkogex #5 March 2, 2018, 2:08pm

> MicahZoltu:
>
> That type of exchange (where all of one asset is traded for all of the other asset at the end of the day) needs a mechanism for limit orders. As a user, I may be interested in liquidating a large amount of asset ABC. I want to be able to put it all on the exchange but specify the exchange rate I’m willing to accept.

Good idea - we will add this!

1 Like

kladkogex #6 March 2, 2018, 4:44pm

> vbuterin:
>
> However, there is a flaw in this design: it is vulnerable to front running attacks.

BTW there is another interesting flaw - lets say there is a bad news and the value of the token drops on external market, by say 20%.

Then, since the money maker follows the old price, the money maker will essentially present a bounty to the first arbitrageurs who comes.

The bounty could be easily $1M

This $1M will be paid to the first arbitrageur, and all of them will rush to the be the first. Then many of them will pay lots of gas and get into the same block. It will then come to a position within a block. Now who is controlling positions within a block? - miners.

So the money maker will provide a source of profits for miners - essentially miners will make lots of money on each fluctuation of the price.

But since the market maker plays a zero-sum game, miners making profits will translate into suboptimal purchases for regular users of the market maker.

4 Likes

vbuterin #7 March 3, 2018, 12:58am

Yep, agree that the scheme absolutely has inefficiencies in that regard.

3 Likes

kladkogex #8 March 9, 2018, 9:25am

Lots of sad sad news this week from US regulators …

First, if you run any type of an exchange **it needs to be registered with SEC**

And then, if you exchange currency for currency you also become **a money transmitter**

There is a question then whether there is any way to run any type of a market maker smart contract legally …

This looks like they are closing pretty much all ways to exchange tokens.

An interesting thing is that Ethereum already has an embedded market that exchanges ETH for gas 🙂 I do not think that SEC guys understand this though 🙂

1 Like

haydenadams #9 March 9, 2018, 3:31pm

Very interesting ideas! I will look into adding one of these solutions to **my implementation** of an x*y=k market maker.

4 Likes

a4nkit #10 March 10, 2018, 8:21pm

I guess the SEC rule is for centralized exchange not decentralized exchange.

1 Like

haydenadams #11 June 12, 2018, 9:12pm

@vbuterin Trying to come up with a better name for this. What do you think about constant product market maker? Or constant product market scoring rule (CPMSR) if fitting in with LMSR is desired

2 Likes

vbuterin #12 June 13, 2018, 8:31am

Sure, either of those sound good to me.

1 Like

k06a #13 June 15, 2018, 12:40pm

@vbuterin recently we won Bankex Foundation hackathon with the idea, based on your original proposition.

We reinforced your idea and aggregated multiple tokens with different weights in one multi-token, which allows anyone to exchange any subtoken for any with the x∗y=k formula. We wanna achieve auto-rebalancing multi-token to represent whole investor token portfolio. And also achieve safe proportions change of balanced multi-token for safe strategy copying by minting and burning funds multi-tokens by providing and extracting subtokens.

Did you think in that direction? Will be great to receive some feedback.

Here are drafts of the smart contracts: **https://github.com/MultiTKN/MultiTKN**
We've started to write our whitepaper: **https://hackmd.io/s/Bks8EfZxX**

1 Like

imkharn #14 August 11, 2020, 4:39pm

Is there any way to undo transaction ordering from miners?

For example, saving all of the requested AMM trades until the next block, then processing them in a random order?

1 Like

Mulitwfn #15 August 27, 2020, 7:40am

This isn't a study to reduce slippage.