A Global Market Maker

Marcelo Bagnulo, Angel Hernando-Veciana, Efthymios Smyrniotis*

Universidad Carlos III de Madrid
{firstname.lastname}@uc3m.es

1 About DEXs & AMMs

As cryptocurrentcies' popularity grows, so it does the need to exchange them for other cryptocurrencies and for fiat currencies. Crypto exchanges have emerged to satisfy these needs. A crypto exchange is a platform that allows users to buy, sell, and trade cryptocurrencies. Crypto exchanges can be centralized or decentralized. Centralized EXchanges (CEXs) are owned and operated by a single entity, which holds users' funds and executes trades. Decentralized EXchanges (DEXs) do not rely on a central authority to hold users' funds or execute trades. Instead, they use smart contracts running over a public blockchain such as Ethereum, to automate the trading process. DEXs have become increasingly popular reaching a daily trading volume of billions of dollars. There are two main types of DEXs, namely, order book DEXs and Automated Market Makers (AMMs). Order book DEXs were introduced first but nowadays AMMs are the most common type of DEX. Order book DEXs are similar to traditional order book exchanges. Users place orders to buy or sell assets, and these orders are matched with each other to execute trades. AMMs work by creating liquidity pools, which are essentially large baskets of assets that are held by the exchange. When a user wants to trade an asset, they do not actually trade with another user. Instead, they trade with the liquidity pool. The AMM will then use a predetermined mathematical formula to determine the price of the trade.

One of the most widely used AMMs is the Constant Product Market Makers (CPMMs). CPMMs were first introduced by Vitalik Buterin in a blog post in 2016. In his post, Buterin argued that CPMMs were a more efficient and decentralized way to provide liquidity on decentralized exchanges than traditional market makers. Hayden Adams firstly implemented CPMMs in his decentralized exchange, Uniswap, in 2017. Uniswap quickly became the most popular decentralized ex- change on Ethereum, and CPMMs became the most popular type of AMM. CPMMs work by maintaining a constant product of the reserves of two assets in a pool. This means that if the amount of one asset in the pool goes up, the amount of the other asset must go down in order to keep the product constant. The formula for a CPMM is:

$$x * y = k \tag{1}$$

* This work was partially funded by the European Union through NGI Sargasso's GMM project under Horizon Europe (Grant Agreement No. 101092887).

where:

- x is the reserve of asset X in the pool
- y is the reserve of asset Y in the pool
- k is a constant

2 Arbitrages & MEV

The widespread adoption of CPMMs has introduced inefficiencies in the cryptocurrency exchange market, including arbitrage opportunities, MEV extraction and impermanent loss as we describe next.

In a market where multiple CPMMs are operating, the fact that each CPMM adjusts its price based only on the trades it executes, implies that, after a set of trades, it is possible that the different CPMMs have different prices. This commonly results in **arbitrage** opportunities i.e. the possibility of making a profit through the acquisition of an asset through one of the CPMMs where the price is lower and the subsequent sale of the asset in another CPMM where the price is higher. In this and all the other examples we do not consider neither AMMs fees nor gas unless explicitly stated.

Toy example 1: Arbitrage between CPMMs. Consider two CPMMs, CPMM1 and CPMM2, each one with a liquidity pool containing 100 ETH and 400,000 UST. The price is the same for both CPMMs and equal to 4,000 UST/ETH. All the computations in this example are done using formula 1.
Suppose that a trade for buying 10 ETH arrives to CPMM1. This means that after the trade, CPMM1's pool will contain 90 ETH and 444,444 UST. The price in CPMM1 is now 4,938 UST/ETH. The trader has paid 44,444 USTs. An arbitrage opportunity has emerged.
The arbitrageur will then buy 5 ETH from CPMM2 and sell 5 ETH to CPMM1. After the arbitrage, the reserves of both CPMMs will be equal to 95 ETH plus 421,052 UST. The arbitrageur has made a profit of 2,339 UST.
Suppose at this point that a trader perform the reverse operation, meaning that it sells 10 ETH to CPMM1 After the trade CPMM1 pool will contain 105 ETH and 380,952 USDT and the trader has obtained 40,100 UST. Again, an arbitrage operation has emerged which is seized by an arbitrageur buying 5 ETH from CPMM1 and selling them to CPMM1, making a profit of 2,005 UST. After these two arbitrage operations, the reserves of both AMMs wil be restored to 100 ETH and 400,000 USDT each.
Overall, in these 6 operations, the CPMMs retained their initial reserves intact, the arbitrageur has made a profit of 4,344 USDT which matches to the net loss of the trader(s)-

MEV (Maximum Extractable Value) refers to the profit that miners, validators or other parties can make by including, excluding, or reordering transactions within a block. MEV can manifest in various ways. Frontrunning and backrun- ning are two common strategies for capturing MEV in CPMMs. In frontrunning, the MEV-extractor observes a pending transaction on the blockchain and quickly executes a similar transaction with higher gas fees, en- suring that their transaction gets prioritized in the next block. Backrunning is the opposite of frontrunning. Instead of preempting pending transactions, the MEV-extractor observes transactions after they have been included in a block but before the block is finalized. They then execute trades to exploit anticipated price movements caused by those transactions. Both frontrunning and backrunning can be jointly utilized in "sandwich at- tacks," where an attacker places their own trades strategically before and after a target transaction. This allows them to profit from the price impact of the target transaction. In CPMMs, sandwich attacks involve inserting trades before and after a large trade to capitalize on the resulting slippage.

MEV (Maximum Extractable Value) refers to the profit that miners, validators or other parties can make by including, excluding, or reordering transactions within a block. MEV can manifest in various ways. Frontrunning and backrunning are two common strategies for capturing MEV in CPMMs.

In frontrunning, the MEV-extractor observes a pending transaction on the blockchain and quickly executes a similar transaction with higher gas fees, ensuring that their transaction gets prioritized in the next block. Backrunning is the opposite of frontrunning. Instead of preempting pending transactions, the MEV-extractor observes transactions after they have been included in a block but before the block is finalized. They then execute trades to exploit anticipated price movements caused by those transactions.

Both frontrunning and backrunning can be jointly utilized in "sandwich attacks," where an attacker places their own trades strategically before and after a target transaction. This allows them to profit from the price impact of the target transaction. In CPMMs, sandwich attacks involve inserting trades before and after a large trade to capitalize on the resulting slippage.

Toy example 2: MEV extraction in CPMM. Consider a CPMM with a liquidity pool containing 100 ETH and 400,00 UST. The marginal price is then 4,000 UST/ETH.
Suppose that a trader submits a transaction for buying 10 ETH. The MEV-extractor performs a sandwich attack by placing a transaction to buy 15 before the victim's transaction is executed and another transaction, in this case selling 15 ETH after the victim's transaction was executed. This would works as follows:
Frontrunning: MEV-extractor transaction 1 that buys 15 ETH. The resulting reserves after this initial transaction are 85 ETH and 470,588

UST. The marginal price price has increased to 5,536 UST/ETH. The MEV-extractor has spent 70,588 UST.
Victim's transaction: THe victim buy 10 ETH. The resulting reserves are then 75 ETH and 533,333 UST. The victim has payed 62,745 USDT i.e. a surplus of 18,301 USDT compared to what she would have payed without the frontrunning.
Backrunning: MEV-extractor transaction 2 that sells 15 ETH. The CPMM's reserves after the trade are 90 ETH and 444,444 UST. The MEV-extractor obtains 88,858 UST, making a total profit of 18,301 USDT (which is exactly what the victim's lost, of course).

$$InitialReserves = 2 * (100ETH * 8,163UST/ETH + 400,000UST)$$
$$FinalReserves = 2 * (70ETH * 8,163UST/ETH + 571,429UST)$$
$$IL = InitialReserves - FinalReserves = 146,922UST$$

3 A naive Global Market Maker

We analyse all these ideas starting with the most obvious observation: why not to determine the prices at each individual AMM as in a hypothetical single AMM that holds the sum of the liquidity pools of all the AMMs. To define such algorithm, we denote by $I \equiv \{1, 2, \ldots, n\}$ the set of AMMs, each with reserves (x_i, y_i), $i \in I$, and where

$$x \equiv \sum_{j \in I} x_j, \text{ and } x_{-i} \equiv \sum_{j \in I \setminus \{i\}} x_j,$$

and where y, y_{-i} are defined similarly. We also use the superscript 0 to denote the initial reserves of AMM i, and $(\Delta x_i, \Delta y_i)$ a trade order submitted to AMM i, where the convention is that $\Delta x_i > 0$ indicates that the AMM buys crypto X. We say that $(\Delta x_i, \Delta y_i)$ is feasible when this is a trade accepted by the algorithm of AMM i.

The *naive Global Market Maker* (nGMM) algoritm trades Δx_i for Δy_i if:

$$(x + \Delta x_i)\,(y + \Delta y) = x \cdot y. \tag{2}$$

Toy example 4: nGMM Consider two market makers GMM1 and GMM2 using nGMM, where the initial distribution of liquidity pools is $(\vec{x}^0, \vec{y}^0) = (100ETH, 100ETH, 400000UST, 400000UST)$. Suppose that GMM1 receives a buy-order of 10 ETH. The resulting reserves of USDT of GMM1 can be computed as $\frac{(100+100)*(400,000+400,00)}{(90+100)} - 400,000$ which is roughly 442,105 UST. This means that trader payed 42,105 USDT to obtain the 10 ETH. This is less than what the trader has to pay in

the CPMM case (see Toy Example 1). Actually, the reduction in what the trader had to pay is exactly the arbitrageur profit computed in Toy Example 1.

The nGMM has the virtue of eliminating the cost of arbitrageur and pass the savings to traders, however it is not a practical implementation because the nGMM is exploitable, as the next example shows.

Toy example 5: Exploiting nGMM Consider two market makers nGMM1 and nGMM2 using nGMM, where the initial distribution of liquidity pools is $(\vec{x}^0, \vec{y}^0) = (100ETH, 100ETH, 400000UST, 400000UST)$.
Exhausting nGMM. Suppose that a nGMM1 receives a transaction buying 100 ETH. nGMM1's reserves in USDT can be computed as $\frac{(100+100)*(400{,}000+400{,}00)}{(0+100)} - 400,000$ which is roughly 1,2M UST. nGMM1's reserves in ETH are zero. This shows how with a finite amount of UST, nGMM1's reserves in ETH can be exhausted.
Exploiting nGMM. We start again with the initial distribution of reserves of 100 ETH and 400,000 USDT in both nGMM1 and nGMM2. Consider the effect of the following 3 transactions:
Exploit transaction 1: Trader buys 10 ETH from nGMM1. The resulting reserves of nGMM1 are 90 ETH and 442,105 UST. The trader has paid 42,105 USDT for the 10 ETH. nGMM2 still has 100 ETH and 400,000 UST.
Exploit transaction 2: Trader buys 10 ETH from nGMM2. The resulting reserves of nGMM2 are then 90 ETH and 446,783 UST. The trader paid 46,784 UST. nGMM2 has more reserves than nGMM1 (same amount of ETH but more UST)
Exploit transaction 3: Trader sells 10 ETH to nGMM1. The reserves for nGMM1 woult then be 100 ETH and 395,321 UST, which is strictly less than the initial reserves of nGMM1 (same amount of ETH and less UST). nGMM has experiences a net loss of reserves. The trader has obtained 46,784 UST.
Observe that the last two transactions cancel out exactly and that the result is a net transfer of reserves from nGMM1 to nGMM2 i.e. the trader does not obtain any profit from the attack). This means that the nGMM approach generates the incentives for participating market makers to attack each other to extract each other reserves.

4 The Global Market Maker

The reason why nGMM is exhaustible is clear. If the prices must react to the relative abundance of the cryptos in the whole set of AMM, it may happen that

an AMM sells all its holdings of a crypto at a finite price. Exploitation of the nGMM is also because the prices reflect scarcity for the whole set of AMM's. Thus, if AMM i receives a sell-order of Δxi units of X, first, and then another AMM receives another sell-order of X, followed by another order to AMM i to buy Δxi units of X, it turns out that AMM i, is buying X when it is relatively scarce in the whole market, and thus it is relatively expensive, and selling it back when it has become relatively more abundant, and thus cheaper. This is the reason why the AMM ends up exploited. propose a modification that avoids these two problems but still eliminate the cost of arbitrage by making prices to depend on the global scarcity of the assets.

These two vulnerabilities render the nGMM unsuitable for practical implementation. Our approach, therefore, is to explore how much we can adjust the CPMM to approximate the nGMM while avoiding these vulnerabilities. We restrict our attention to algorithms whose terms of trade lie between those of the CPMM and the nGMM. The restriction to algorithms between the CPMM and the nGMM should be interpreted as a minimal departure from existing practices. The CPMM represents the current state of the art, while the nGMM serves as its ideal generalization to mitigate the inefficiencies caused by market fragmentation. Our objective is to approximate the nGMM as closely as possible while deviating as little as possible from the CPMM. To do so, the GMM will determine the prices as the minimum proces offered by the CPMM and the nGMM. The Global Market Maker (GMM) is therefore defined by:

$$\Delta y_{GMM}(\Delta x; x, y) = \min\{\Delta y_{CPMM}(\Delta x; x_1, y_1), \Delta y_{nGMM}(\Delta x; x, y)\} \quad (3)$$

By exploiting efficiently the aggregation of the information from the liquidity pools of the other AMMs, the GMM improves in the following key dimensions of the design of an AMM: first, it eliminates arbitrage opportunities and splits the efficiency gains between traders and AMMs; second, it reduces slippage and thus the profitability of MEV sandwich trades.

Toy example 6: GMM
In this example, we show that under the same conditions of the example 1, in GMM no arbitrage opportunities appear, and that the profit of the arbitrageurs is split between the AMMs and the traders.
We start with two GMMs (GMM1 and GMM2) that have initial reserves of 100 ETH and 400,000 USDT each.
Similarly to Example 1, a trader buys 10 ETH from GMM1. Since both GMMs have the same reserves, GMM1 uses the CPMM formula for this trade (i.e. same as Example 1). After this trade, GMM1's pool contains 90 ETH and 444,444 USDT and GMM2's pool is unchanged. The trader has paid 44,444 UST. The resulting prices are as follows:

GMM1 Buy ETH	GMM1 Sell ETH	GMM2 Buy ETH	GMM2 Sell ETH
4,938 UST/ETH	4,444 UST/ETH	4,444 UST/ETH	4,000 UST/ETH

It follows that no arbitrage opportunities emerge, as the most convenient price to sell and buy from the GMMs is the common price.
Suppose now that, similarly to Example 1, the reverse transaction is executed, namely that a trader sells 10 ETH to GMM1. In this case, the GMM determines the prices in the convergent region. This implies that resulting reserves of GMM1 after the trade are 100 ETH and 402,222 USDT and that the trader obtained 42,222 UST.
This means that after this trade, the reserves of the GMM have increased in 2,222 USDT and that the trader have paid 2,122 USDT LESS than in Example 1. In other words, GMM splits almost evenly the profit of the arbitrageur between the trader and the GMM.

Toy example 7: MEV extraction in GMM
Consider 2 GMMs (GMM1 and GMM2) with initial reserves of 100 ETH and 400,000 USDT each. Suppose that a trader issues an order to buy 10 ETH 10 from GMM1 and that the MEV extractor launches a sandwich attack of 15 ETH. The transactions of the MEV-extractor and victim will be processed as follows:
Frontrunning: MEV-extractor transaction 1 that buys 15 ETH. The resulting reserves after this initial transaction are 85 ETH and 470,588 UST. The marginal price price has increased to 5,536 UST/ETH. The MEV-extractor has spent 70,588 UST. This is the same as Example 2.
Victim's transaction: The victim buy 10 ETH. The resulting reserves are then 75 ETH and 533,333 UST. The victim has payed 62,745 USDT i.e. a surplus of 18,301 USDT compared to what she would have payed without the frontrunning. Again, this is the same as Example 2.
Backrunning. The MEV-extractor sells the 15 ETH. In this case, GMM1 uses the convergent formula, which yields that the MEV-extractor will obtain 73,684 UST, making a net profit of 3,096 UST, which is 15,205 USDT less than in Example 2.

5 Conclusions

We have examined how fragmented liquidity across Automated Market Makers (AMMs) inflates trading costs through arbitrage, slippage, and MEV-driven

front-running. We introduced the *Global Market Maker* (GMM), an AMM algorithm that incorporates blockchain-wide reserve information yet retains the safety profile of the conventional Constant-Product rule. By quoting the smaller of the two amounts a trader would pay under the local or the global reserve view, GMM aligns swap prices with aggregate scarcity while ensuring that no pool can be emptied at a finite cost.

We show that GMM (i) removes systematic arbitrage, reallocating the former arbitrage rent between traders and liquidity providers, and (ii) lowers slippage, thereby reducing the profitability of sandwich attacks. These properties indicate that policy-driven price manipulation becomes less attractive, which in turn should foster deeper liquidity and more stable markets.

GMM can be deployed as a stand-alone smart-contract module that queries an oracle for global reserve data, allowing incremental adoption without altering custody or governance models. Future research will formalise incentive compatibility under heterogeneous liquidity-provider preferences, extend the design to multi-asset pools, and conduct live-network pilots to measure end-to-end effects on user welfare and liquidity provision.