Arbitrageurs' profits, LVR, and sandwich attacks: batch trading as an AMM design response

Andrea Canidio (CoW Protocol)
joint work with Robin Fritsch (ETH Zurich)

Blockchain@X - OMI Workshop

How to design blockchain-based financial markets?

Dominant model: Constant Function Automated Market Makers

How to design blockchain-based financial markets?

Dominant model: Constant Function Automated Market Makers

- Characterized by its liquidity reserves X (e.g., ETH) and Y (e.g., USDC) and its invariant function $f(.,.) : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$.

How to design blockchain-based financial markets?

Dominant model: Constant Function Automated Market Makers

- Characterized by its liquidity reserves X (e.g., ETH) and Y (e.g., USDC) and its invariant function $f(.,.) : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$.
- There is a swap proposed: purchase x ETH from the AMM in exchange for y USDC.

How to design blockchain-based financial markets?

Dominant model: Constant Function Automated Market Makers

- Characterized by its liquidity reserves X (e.g., ETH) and Y (e.g., USDC) and its invariant function $f(.,.) : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$.
- There is a swap proposed: purchase x ETH from the AMM in exchange for y USDC.
- accept iff

$$f(X, Y) = f(X - x, Y + y)$$

How to design blockchain-based financial markets?

Dominant model: Constant Function Automated Market Makers

- Characterized by its liquidity reserves X (e.g., ETH) and Y (e.g., USDC) and its invariant function $f(.,.) : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$.
- There is a swap proposed: purchase x ETH from the AMM in exchange for y USDC.
- accept iff

$$f(X, Y) = f(X - x, Y + y)$$

- The invariant function must satisfy path-independence

How to design blockchain-based financial markets?

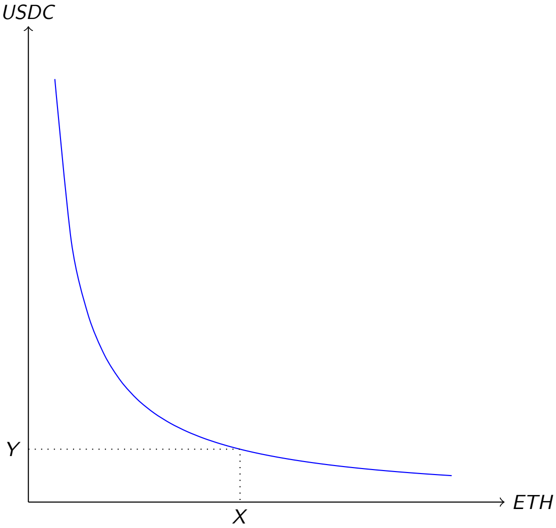
Dominant model: Constant Function Automated Market Makers

- Characterized by its liquidity reserves X (e.g., ETH) and Y (e.g., USDC) and its invariant function $f(.,.) : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$.
- There is a swap proposed: purchase x ETH from the AMM in exchange for y USDC.
- accept iff

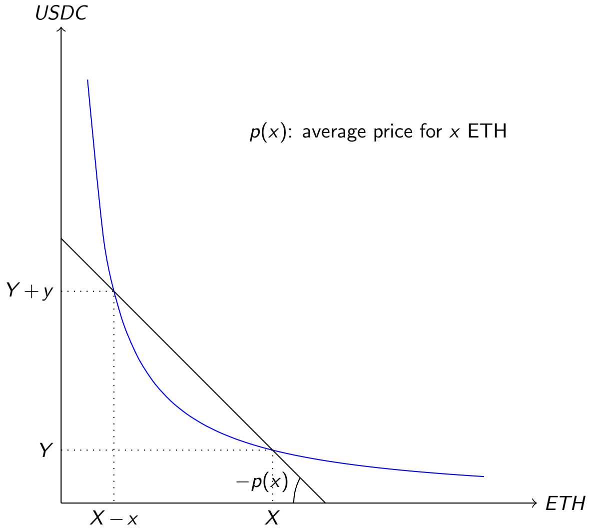
$$f(X, Y) = f(X - x, Y + y)$$

- The invariant function must satisfy path-independence
- Most common invariant function: the product $\Rightarrow$ $p(x) = y/x = Y/(X - x)$

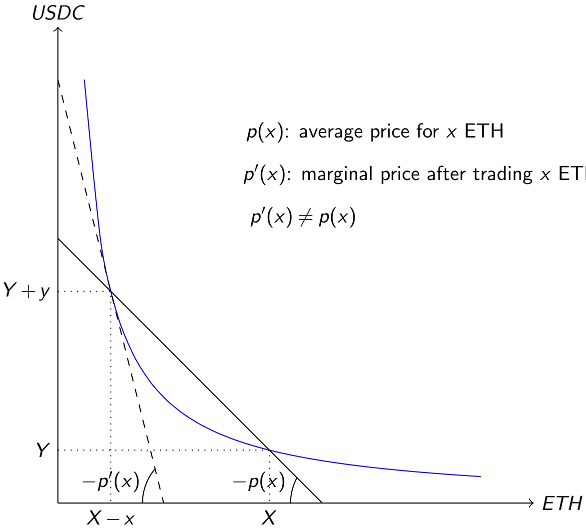
Constant Product Automated Market Maker



Constant Product Automated Market Maker



Constant Product Automated Market Maker



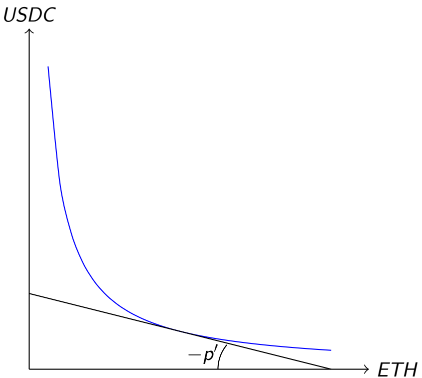
$p(x)$: average price for x ETH

$p'(x)$: marginal price after trading x ETH

$p'(x) \neq p(x)$

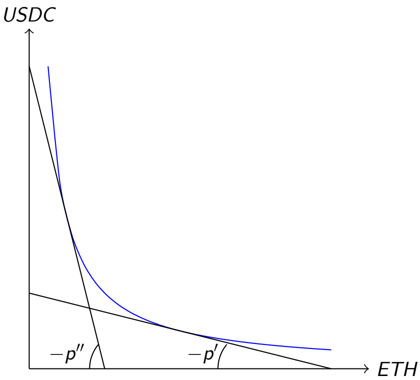
Problem with Constant function AMM: (1) Arbitrage Profits (aka Loss-vs-rebalancing, aka LVR)

- The equilibrium price of ETH (in USDC) is determined on Binance. Initially, it is p' and is equal to the AMM marginal price.



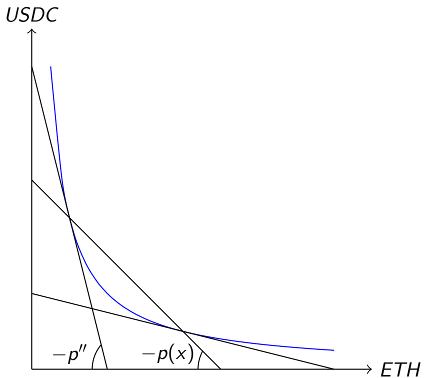
Problem with Constant function AMM: (1) Arbitrage Profits (aka Loss-vs-rebalancing, aka LVR)

- The equilibrium price of ETH (in USDC) is determined on Binance. Initially, it is p' and is equal to the AMM marginal price.
- The equilibrium price increases to p''. Arbitrageurs rebalance the AMM until its marginal price equals p''



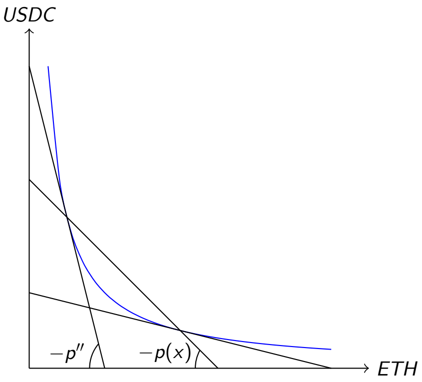
Problem with Constant function AMM: (1) Arbitrage Profits (aka Loss-vs-rebalancing, aka LVR)

- The equilibrium price of ETH (in USDC) is determined on Binance. Initially, it is p' and is equal to the AMM marginal price.
- The equilibrium price increases to p''. Arbitrageurs rebalance the AMM until its marginal price equals p''
- The average price for the rebalancing trade is less than p'' $\rightarrow$ the first arbitrageur reaching the AMM makes a profit (at the expense of the LP)



Problem with Constant function AMM: (1) Arbitrage Profits (aka Loss-vs-rebalancing, aka LVR)

- The equilibrium price of ETH (in USDC) is determined on Binance. Initially, it is p' and is equal to the AMM marginal price.
- The equilibrium price increases to p''. Arbitrageurs rebalance the AMM until its marginal price equals p''
- The average price for the rebalancing trade is less than p'' $\rightarrow$ the first arbitrageur reaching the AMM makes a profit (at the expense of the LP)
- Defensive mechanism: fees



Problem with Constant Function AMM: (2) Sandwich attacks

- Fact (1): reordering of transactions is possible
- Fact (2): by default, pending transactions are public

If someone sends a transaction to purchase x ETH from an AMM. An attacker can:

Problem with Constant Function AMM: (2) Sandwich attacks

- Fact (1): reordering of transactions is possible
- Fact (2): by default, pending transactions are public

If someone sends a transaction to purchase x ETH from an AMM. An attacker can:

- front run the victim with the same trade (also buy ETH) $\rightarrow$ increase the price of ETH

Problem with Constant Function AMM: (2) Sandwich attacks

- Fact (1): reordering of transactions is possible
- Fact (2): by default, pending transactions are public

If someone sends a transaction to purchase x ETH from an AMM. An attacker can:

- front run the victim with the same trade (also buy ETH) $\rightarrow$ increase the price of ETH
- back run the victim with the opposite trade (sell ETH)

Problem with Constant Function AMM: (2) Sandwich attacks

- Fact (1): reordering of transactions is possible
- Fact (2): by default, pending transactions are public

If someone sends a transaction to purchase x ETH from an AMM. An attacker can:

- front run the victim with the same trade (also buy ETH) $\rightarrow$ increase the price of ETH
- back run the victim with the opposite trade (sell ETH)
- $\Rightarrow$ the attacker buys cheap and sells expensive; the victim buys expensive

Our paper: batching trades (and designing the AMM around this) solves both problems

- All trades are collected off-chain and batched:
    - they are settled p2p if possible; the remaining is settled on an AMM (at the same price)
    - the AMM can be accessed only via the batch

Our paper: batching trades (and designing the AMM around this) solves both problems

- All trades are collected off-chain and batched:
  - they are settled p2p if possible; the remaining is settled on an AMM (at the same price)
  - the AMM can be accessed only via the batch
- No need to satisfy path independence: It is possible to design a "Function Maximizing" automated market maker (FM-AMM)

Our paper: batching trades (and designing the AMM around this) solves both problems

- All trades are collected off-chain and batched:
  - they are settled p2p if possible; the remaining is settled on an AMM (at the same price)
  - the AMM can be accessed only via the batch
- No need to satisfy path independence: It is possible to design a "Function Maximizing" automated market maker (FM-AMM)
- *Theory*: if there is a large external trading venue (where the price is determined) and arbitrageurs, **FM-AMM always trades at the equilibrium price**
  - Arb profits (LVR) and sandwich attacks are eliminated

Our paper: batching trades (and designing the AMM around this) solves both problems

- All trades are collected off-chain and batched:
  - they are settled p2p if possible; the remaining is settled on an AMM (at the same price)
  - the AMM can be accessed only via the batch
- No need to satisfy path independence: It is possible to design a "Function Maximizing" automated market maker (FM-AMM)
- *Theory*: if there is a large external trading venue (where the price is determined) and arbitrageurs, **FM-AMM always trades at the equilibrium price**
  - Arb profits (LVR) and sandwich attacks are eliminated
- *Empirical exercise*: Using price data, we compare returns to providing liquidity to Uniswap v3 to a simulated FM-AMM with no noise traders

Our paper: batching trades (and designing the AMM around this) solves both problems

- All trades are collected off-chain and batched:
  - they are settled p2p if possible; the remaining is settled on an AMM (at the same price)
  - the AMM can be accessed only via the batch
- No need to satisfy path independence: It is possible to design a "Function Maximizing" automated market maker (FM-AMM)
- *Theory*: if there is a large external trading venue (where the price is determined) and arbitrageurs, **FM-AMM always trades at the equilibrium price**
  - Arb profits (LVR) and sandwich attacks are eliminated
- *Empirical exercise*: Using price data, we compare returns to providing liquidity to Uniswap v3 to a simulated FM-AMM with no noise traders

Our paper: batching trades (and designing the AMM around this) solves both problems

- All trades are collected off-chain and batched:
  - they are settled p2p if possible; the remaining is settled on an AMM (at the same price)
  - the AMM can be accessed only via the batch
- No need to satisfy path independence: It is possible to design a "Function Maximizing" automated market maker (FM-AMM)
- *Theory*: if there is a large external trading venue (where the price is determined) and arbitrageurs, **FM-AMM always trades at the equilibrium price**
  - Arb profits (LVR) and sandwich attacks are eliminated
- *Empirical exercise*: Using price data, we compare returns to providing liquidity to Uniswap v3 to a simulated FM-AMM with no noise traders $\Rightarrow$ they are very similar.

The Function Maximizing AMM (FM-AMM) with product function

FM-AMM is a price-taking agent that trades to maximize the product of its liquidity reserves subject to a budget constraint:

The Function Maximizing AMM (FM-AMM) with product function

FM-AMM is a price-taking agent that trades to maximize the product of its liquidity reserves subject to a budget constraint:

$$x^{FM-AMM}(p) = \text{argmax}_x \{(X - x)(Y + p \cdot x)\}.$$

$$x^{FM-AMM}(p) = \frac{1}{2}\left(X - \frac{Y}{p}\right).$$

The Function Maximizing AMM (FM-AMM) with product function

FM-AMM is a price-taking agent that trades to maximize the product of its liquidity reserves subject to a budget constraint:

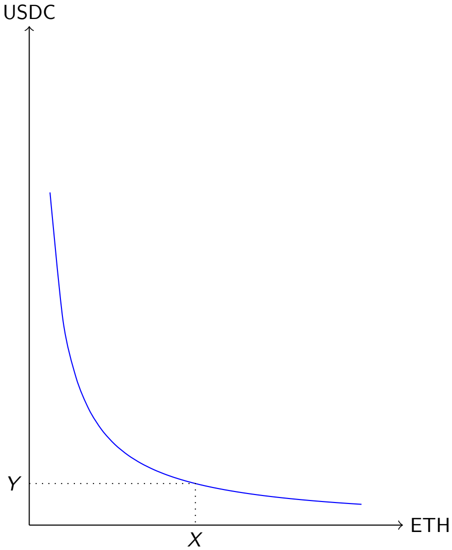
$$x^{FM-AMM}(p) = \text{argmax}_x \{(X - x)(Y + p \cdot x)\}.$$

$$x^{FM-AMM}(p) = \frac{1}{2}\left(X - \frac{Y}{p}\right).$$

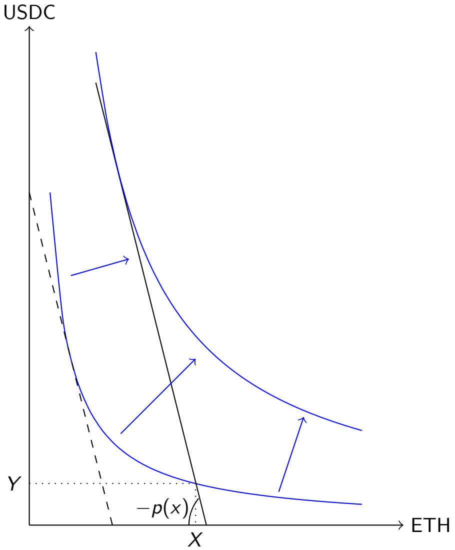
Hence, to purchase x ETH on the FM-AMM, the price needs to be:

$$p^{FM-AMM}(x) = \frac{Y}{X - 2x}.$$

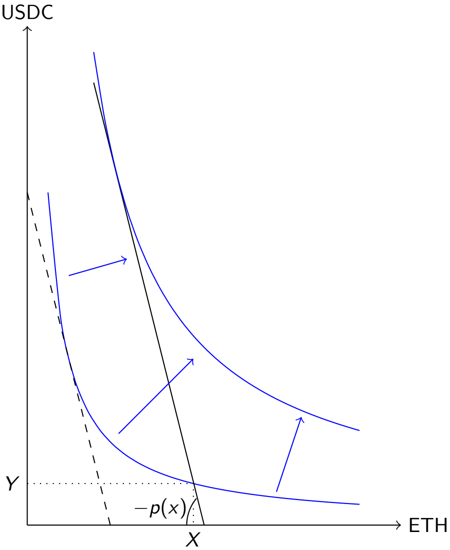
The FM-AMM "moves up the curve"



The FM-AMM "moves up the curve"

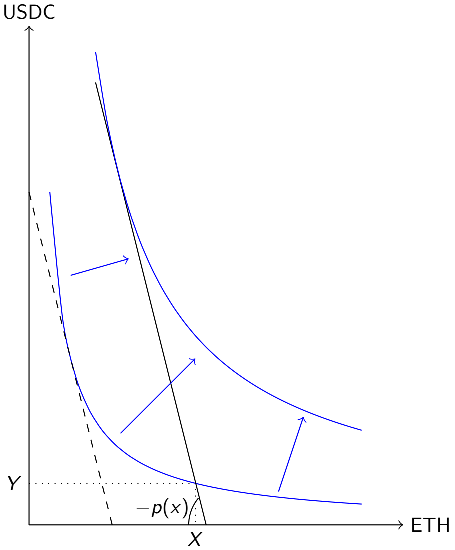


The FM-AMM “moves up the curve”



- FM-AMM is **clearing-price consistent**: the price at which it trades equals the marginal price after the trade

The FM-AMM "moves up the curve"



- FM-AMM is **clearing-price consistent**: the price at which it trades equals the marginal price after the trade
- FM-AMM violates path independence: it can be exploited by splitting trades → **batching is necessary**

Theoretical model: FM-AMM in equilibrium

- An FM-AMM and a large off-chain trading venue where the equilibrium price p^* is determined

Theoretical model: FM-AMM in equilibrium

- An FM-AMM and a large off-chain trading venue where the equilibrium price p^* is determined
- Noise traders trading on the FM-AMM, price-taking arbitrageurs trading both on FM-AMM and the large off-chain trading venue

Theoretical model: FM-AMM in equilibrium

- An FM-AMM and a large off-chain trading venue where the equilibrium price p^* is determined
- Noise traders trading on the FM-AMM, price-taking arbitrageurs trading both on FM-AMM and the large off-chain trading venue
- Time is discrete:
  - *Even periods* = different block (on-chain transaction occurs)
  - *Odd periods* (off-chain events):

    (1) the equilibrium price is determined in the large off-chain venue;

    (2) traders submit orders for inclusion in the batch (to be executed in the next even period)

Theoretical model: FM-AMM in equilibrium

- An FM-AMM and a large off-chain trading venue where the equilibrium price p^* is determined
- Noise traders trading on the FM-AMM, price-taking arbitrageurs trading both on FM-AMM and the large off-chain trading venue
- Time is discrete:
  - *Even periods* = different block (on-chain transaction occurs)
  - *Odd periods* (off-chain events):
    (1) the equilibrium price is determined in the large off-chain venue;
    (2) traders submit orders for inclusion in the batch (to be executed in the next even period)
- FM-AMM charges no fee for inclusion in a batch, and a fee τ (in the input token) for settling an order on the FM-AMM

Theoretical model: FM-AMM in equilibrium

Proposition:

Suppose that, at the end of an even period, the reserves of the FM-AMM are X and Y. In the equilibrium of the subsequent odd period, after p^* is realized, if noise traders collectively submit trade A to the batch, then arbitrageurs will collectively submit trade $y(p^*)$ such that

$$\tilde{p}(A + y(p^*), \tau) = p^*$$

where $\tilde{p}(A + y(p^*), \tau)$ is the FM-AMM effective price (i.e., price after fees).

Theoretical model: FM-AMM in equilibrium

Proposition:

Suppose that, at the end of an even period, the reserves of the FM-AMM are X and Y. In the equilibrium of the subsequent odd period, after p^* is realized, if noise traders collectively submit trade A to the batch, then arbitrageurs will collectively submit trade $y(p^*)$ such that

$$\tilde{p}(A + y(p^*), \tau) = p^*$$

where $\tilde{p}(A + y(p^*), \tau)$ is the FM-AMM effective price (i.e., price after fees).

- No losses to arbitrageurs (no LVR)
- No sandwich attack

Theoretical model: FM-AMM in equilibrium

Proposition:

Suppose that, at the end of an even period, the reserves of the FM-AMM are X and Y. In the equilibrium of the subsequent odd period, after p^* is realized, if noise traders collectively submit trade A to the batch, then arbitrageurs will collectively submit trade $y(p^*)$ such that

$$\tilde{p}(A + y(p^*), \tau) = p^*$$

where $\tilde{p}(A + y(p^*), \tau)$ is the FM-AMM effective price (i.e., price after fees).

- No losses to arbitrageurs (no LVR)
- No sandwich attack
- p^* determines the rebalancing trade, which determines the fees earned

Theoretical model: FM-AMM, CFAMM and risk

	CFAMM	FM-AMM
AMM function		
Value of reserves		

Theoretical model: FM-AMM, CFAMM and risk

	CFAMM	FM-AMM
AMM function	risk neutral	
Value of reserves		

Theoretical model: FM-AMM, CFAMM and risk

	CFAMM	FM-AMM
AMM function	risk neutral	risk loving
Value of reserves		

Theoretical model: FM-AMM, CFAMM and risk

	CFAMM	FM-AMM
AMM function	risk neutral	risk loving
Value of reserves	risk averse	

Theoretical model: FM-AMM, CFAMM and risk

	CFAMM	FM-AMM
AMM function	risk neutral	risk loving
Value of reserves	risk averse	risk neutral

Empirical exercise

- Collect price data from Binance (October 2022 to March 2023) for $ETH - USDT$, $BTC - USDT$, $BTC - ETH$.

Empirical exercise

- Collect price data from Binance (October 2022 to March 2023) for $ETH - USDT$, $BTC - USDT$, $BTC - ETH$.
- Use the proposition to simulate how arbitrageurs would rebalance FM-AMM (had it existed) and hence the return providing liquidity to an FM-AMM.

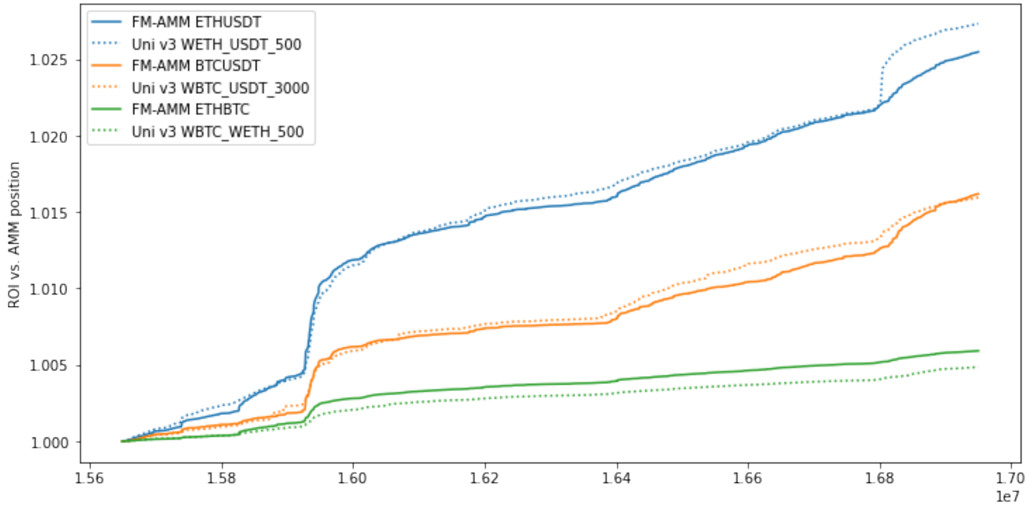
Empirical exercise

- Collect price data from Binance (October 2022 to March 2023) for $ETH - USDT$, $BTC - USDT$, $BTC - ETH$.
- Use the proposition to simulate how arbitrageurs would rebalance FM-AMM (had it existed) and hence the return providing liquidity to an FM-AMM.
- Compute the return of liquidity providers on Uniswap v3 (the leading AMM) over the same period for the pools WETH-USDT (with fee 0.05%), WBTC-USDT (with fee 0.3%), and WBTC-WETH (with fee 0.05%).
    - we use data on the distribution of liquidity at the end of each block and the total fees earned during that block

Empirical exercise

- Collect price data from Binance (October 2022 to March 2023) for $ETH - USDT$, $BTC - USDT$, $BTC - ETH$.
- Use the proposition to simulate how arbitrageurs would rebalance FM-AMM (had it existed) and hence the return providing liquidity to an FM-AMM.
- Compute the return of liquidity providers on Uniswap v3 (the leading AMM) over the same period for the pools WETH-USDT (with fee 0.05%), WBTC-USDT (with fee 0.3%), and WBTC-WETH (with fee 0.05%).
  - we use data on the distribution of liquidity at the end of each block and the total fees earned during that block
- Compare the two.

FM-AMM vs Uniswap v3 pool



FM-AMM vs Uniswap v3 pool

- difference in the total return is: -0.22% (for the ETH-USDT pair), 0.03% (for the BTC-USDT pair) and 0.11% (for the ETH-BTC pair).
- maximum difference in value between the two liquidity positions (expressed in percentage of the initial liquidity position) is 0.30% (for the ETH-USDT pair), 0.14% (for the BTC-USDT pair) and 0.12% (for the ETH-BTC pair).

FM-AMM vs Uniswap v3 pool

- difference in the total return is: -0.22% (for the ETH-USDT pair), 0.03% (for the BTC-USDT pair) and 0.11% (for the ETH-BTC pair).
- maximum difference in value between the two liquidity positions (expressed in percentage of the initial liquidity position) is 0.30% (for the ETH-USDT pair), 0.14% (for the BTC-USDT pair) and 0.12% (for the ETH-BTC pair).
- **The differences in returns are small**.

Conclusions

- Batching allows for a novel AMM design that eliminates arbitrageurs' profits (LVR) and sandwich attacks.
- (for the period and the token pair we consider) for liquidity providers, an FM-AMM that does not earn fees from noise traders performs as well as Uniswap v3
    - An FM-AMM that also earns fees from noise traders should perform better

Literature

- Batch auction: Budish, Cramton, and Shim (2015)
- Sandwich attacks: Park (2022), Torres at al. (2021), Qin et al. (2022).
- Arb profits and LVR: Aoyagi (2020), Capponi and Jia (2021), and Milionis et al. (2022), Milionis et al. (2023)
- surplus maximizing AMM / axiomatization of AMM: Goyal et al. (2022), Schlegel and Mamageishvili (2022)
- several blog posts

Thank you!