Consistency of Automated Market Makers

Vincent Danos
CNRS, France
DI ENS, INRIA, PSL, Paris, France

Weijia Wang
ENS, Paris, France

Abstract

Decentralised Finance has popularised Automated Market Makers (AMMs), but surprisingly little research has been done on their consistency. Can a single attacker extract risk-free revenue from an AMM, regardless of price or other users' behaviour? In this paper, we investigate the consistency of a large class of AMMs, including the most widely used ones, and show that consistency holds.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design

Keywords and phrases Automated Market Makers, Decentralised Finance

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2022.4

Acknowledgements The first author wishes to thank Jérôme de Tichey for introducing him to the problem of consistency, Guillaume Terradot and Hamza El Khalloufi for numerous conversations.

1 Introduction

Blockchains offer in principle a neutral computational medium, where anyone can deploy and interact with smart contracts without interference from external parties. It has been long thought that the ability for parties to enter into interactions that have to follow rules captured un-ambiguously by code could change finance [8]. Indeed, there is now an emerging domain known as decentralized finance (DeFi) re-defining financial primitives and functionalities using smart contracts. Its progresses and potentials were recently recognised in a report of the IMF.[1] One key component of any financial system is the mechanism for matching participants willing to trade. DeFi has brought to the fore a novel class of protocols, namely automated market makers (AMMs), to perform this task. In this paper, we investigate the consistency of such AMMs.

An automated market maker (AMM) is a specific type of decentralized exchange, ie a market place which is fully automatised and implemented as a smart contract. An AMM protocol typically maintains reserves, also referred to as pools, of different assets and employs mathematical algorithms to determine the price of assets and identify which trades it is willing to execute with a particular trader. The AMM pools are populated by users known as liquidity providers (LPs). In return for providing liquidity, LPs receive LP tokens, also referred to as pool tokens, representing their ownership fraction of the pools, and receive accordingly a fraction of the fees paid by traders on each trade. Nothing prevents a user from taking on both roles, that is to say, to deposit/withdraw assets as an LP, while at the same time trading assets with the AMM.

[1] "By taking innovation to a new level [...] DeFi has had extraordinary growth in the past two years, potentially offering higher efficiency and investment opportunities." "DeFi offers broad access to players of any size and has no need for custodian service, potentially improving efficiency and financial inclusion." (IMF report, Apr 2022)

The top-performing AMMs in 2022 achieved weekly trading volumes in the billions of euros [3]. Despite those substantial volumes, little is known about the consistency of the underpinning mathematical mechanisms. Of course, providing liquidity can lead to losses depending on the evolution of prices. In the context of AMMs, loss due to price movement is called impermanent loss. However, in this paper, we address a different risk which is unrelated to price action, arbitrage (exploiting price discrepancies on different markets), or the exploitation of other players' moves (as in various types of front-running). We ask whether a single attacker with unbounded capital can initiate a sequence of interactions with a given AMM, which would lead to a risk-free and price-independent profit. If there is no such sequence, regardless of the initial state of the AMM, then we say the AMM is consistent. There are examples of such attacks, making the question of consistency a practically important one [4].

The latest and only known result on the consistency problem, published in 2020, showed that under reasonable conditions on the AMM mechanics, the AMM is consistent as long as the attacker is only allowed to trade (and not to provide liquidity) [2]. This weaker notion of consistency can be readily proved for a large class of AMMs. It leaves open the question of consistency where the attacker is allowed to combine LP actions and trading.

Outline

We begin with the definition of a large class of AMMs which we call price machines, and narrow down their definition to the case of a single attacker (§2). We turn to the definition of consistency and prove our main abstract result which gives a simple sufficient condition for the consistency of a price machine (§3). The third part of the paper (§4) is devoted to applications. We establish the consistency of DeFi's most popular AMMs: Uniswap [1], Balancer [7], and both versions of the Curve AMM [5, 6].

2 Basic definitions

Throughout the paper we use the terms assets and tokens as synonymous. We call market participants simply users. In this section, we define and discuss *price machines* which form the class of AMMs we investigate.

2.1 Preliminaries

The product ordering on $\mathbb{R}^n$, $n > 0$, is written $\preceq$. It is defined as usual as $v \preceq v'$ iff $v_i \leq v'_i$ for $1 \leq i \leq n$. The strict version is written $\prec$. A relevant intuition for the product ordering is that $v \preceq v'$ iff $p^T(v' - v) \geq 0$ for any "price" vector p with positive coordinates. The positive part of a vector $v \in \mathbb{R}^n$, written v_+, is defined as $(v_+)_i = \max(v_i, 0)$, for $1 \leq i \leq n$. We write $\mathbf{1}$ for the vector with all components equal to 1.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be non-decreasing (increasing) if it preserves the (strict) product ordering.

It is easy to see that a function $f : \mathbb{R}^n \to \mathbb{R}$ is increasing iff it is locally increasing, meaning it preserves $\prec$ on a neighbourhood of each point.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called homogenous of degree $k \geq 0$ if for every $v \in \mathbb{R}^n$ and $s \neq 0$, $f(sv) = s^k f(v)$. It is called positively homogenous if the identity holds for $s > 0$.

We write $\mathbb{R}^*_+$ for the set of positive reals.

2.2 Trading functions

A *trading function* on $n \geq 1$ tokens is an increasing function $\psi : (\mathbb{R}_+^*)^n \to \mathbb{R}_+^*$. We say ψ is 1-homogenous if for $s > 0$, $\psi(sv) = s\psi(v)$.

For a vector of reserves $R \in (\mathbb{R}_+^*)^n$, $\psi(R) \in \mathbb{R}_+^*$ specifies the total amount of LP tokens currently distributed (up to a positive constant multiplicative factor). We also pick ϵ a (small) constant $0 \leq \epsilon \leq 1$ specifying the LP fees.

Given ψ, ϵ, we define a notion of AMM trading on n tokens.

Specifically, we distinguish two types of transitions: *Swaps* and *Transfers*. Both types take current reserves R to new reserves R'. Swaps correspond to trade events, transfers correspond to liquidity provision events (deposits and withdrawals).

Swaps: a swap $R \to^s R'$ must satisfy $\psi(R) = \psi(R' - \epsilon(R' - R)_+)$

The vector $(R' - R)_+$ is the amount of tokens received by the AMM, while the vector $(R - R')_+$ is the amount paid out to the trader. As ψ is increasing, it must be that $\psi(R) \leq \psi(R')$. The excess amount $\psi(R') - \psi(R)$ of LP tokens is divided between LPs in proportion to their current amounts of LP tokens.[2]

When $\epsilon = 0$ the swap constraint becomes simply $\psi(R) = \psi(R')$. This is the reason ψ is often referred to as the *invariant* of the AMM. For (small) $\epsilon > 0$, swaps induce a (small) increase of the invariant. In the limit case $\epsilon = 1$, $R' - \epsilon(R' - R)_+ = R \wedge R'$, hence the constraint can be rewritten $\psi(R) = \psi(R \wedge R')$, which implies $R = R \wedge R'$ because ψ in increasing, or equivalently $R' \succeq R$. In words, the AMM never pays out anything on a swap.

Transfers: a transfer $R \to^t R'$ must satisfy either $R \preceq R'$ (deposit), or $R \succeq R'$ (withdrawal).

Deposits (withdrawals) increase (decrease) the current value of the invariant. The excess amount $\psi(R') - \psi(R)$ of LP tokens (negative in the case of a withdrawal) is given to (taken from) the user who initiated the transfer.

A transfer is said to be *balanced* if there exists $\lambda \in \mathbb{R}_+$ such that $R' = \lambda R$, and *perfectly balanced* if $\psi(\lambda R) = \lambda\psi(R)$.

In the next section, we show (Prop. 4) that, if ψ is homogenous, imbalanced transfers can be decomposed as a swap without fees followed by a balanced transfer. Imbalanced transfers are best understood as a compound transition which is convenient to users.[3]

2.3 Example (verified on-chain [10])

Uniswap v2 is a two-token AMM with trading function $\psi(R) := (R_1 R_2)^{\frac{1}{2}}$, and fee $\epsilon = 0.003$. Say the tokens are named A and B. We start in the following configuration:

- The AMM's pool has $5A + 20B$ tokens.
- Alice has 10 LP tokens and is the only LP, since $\psi(5, 20) = 10$
- Bob has $50A + 200B$ tokens
- Charlie has $10A$ tokens

[2] In other words, no-one gets diluted. In reality AMMs do not literally offer LP tokens to their LPs at each swap. For reasons of efficiency, they keep track of the total number of LP tokens separately, to preserve the proportions of LP tokens for each user on swaps, and determine transfers appropriately.

[3] Indeed, in practice, imbalanced transfers are subject to a fee, In the words of the Balancer AMM documentation "Since Balancer allows for depositing and withdrawing liquidity to Balancer pools using only one of the tokens present in the pool, this could be used to do the equivalent of a swap: provide liquidity depositing token A, and immediately withdraw that liquidity in token B. Therefore a swap fee has to be charged, proportional to the tokens that would need to be swapped for an all-asset deposit." (See single asset deposit withdrawal.)

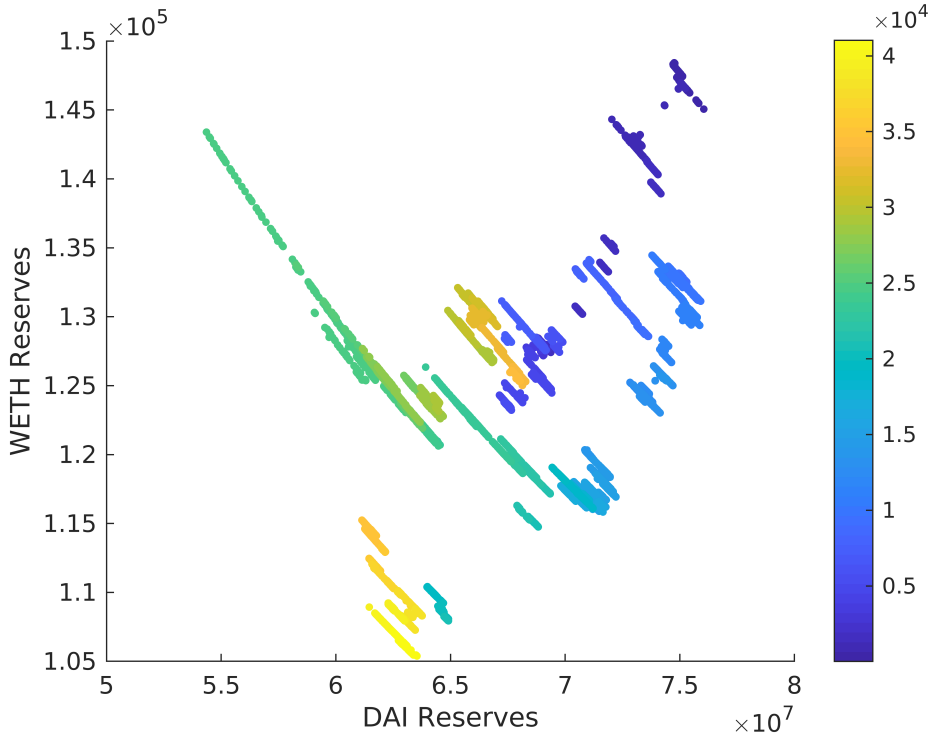

Figure 1 A sequence of aout 40,000 transitions (recorded during the first two weeks of November 2021) on the WETH/DAI Uniswap v2 Ethereum market; colours represent succession in time.

We want to illustrate the various types of transitions:

- (Transfer) Bob deposits $50A + 200B$ tokens (balanced deposit, $\lambda = 11$).
  The pool has now $55A + 220B$ tokens, and Bob receives $\sqrt{50 \cdot 200} = 100$ LP tokens.
- (Swap) Charlie swaps As for Bs, using $10A$ tokens.
  The pool has now $65A$ tokens and $\frac{55 \cdot 220}{65 - 10\epsilon} = 186.239803\,B$ tokens, while Bob receives $220 - 186.239803 = 33.760197\,B$ tokens.
  The excess amount of LP tokens generated by Charlie's swap is shared proportionally, so that Alice has now $\frac{10}{110}\sqrt{65 \cdot 186.239803} = 10.002308$ LP tokens, and Bob has now 100.023085 LP tokens.
- (Transfer) Alice withdraws all her 10.002308 LP tokens (balanced withdrawal)
  Alice receives $\frac{10.002308}{\sqrt{65 \cdot 186.239803}} = \frac{10}{110}$ from the pool, i.e. $\frac{10}{110} \cdot 65 = 5.909091\,A$ tokens and $\frac{10}{110} \cdot 186.239803 = 16.930891 B$ tokens.
  The pool ends up with $59.090909\,A + 169.308912\,B$ tokens.

Fig. 1 gives a historical example (sampled during Nov 2021) of a far longer sequence (of circa 40,000 transitions) on a Uniswap v2 market (on the `ETH/DAI` token pair). Swaps correspond to motions along the hyperbolic contour lines of the trading function. Transfers correspond to jumps from one contour line to a lower (higher) one if a withdrawal (deposit).

2.4 Price machines

The simple example above shows that to properly record the effect of a sequence of transitions on an AMM one needs to incorporate in its state two additional components:

- the (proportions of) LP tokens held by each user,
- as well as their own reserves in the tokens of interest.

To keep things simple we now suppose $\epsilon = 0$. (Fees can be computed for each swap a posteriori anyways.) The resulting model is defined below as a labelled transition system, with its state space, labels, and labelled transitions.

▶ **Definition 1.** *A price machine with n-token trading function ψ and user set U is a labelled transition system with:*

- *state space* $S = (\mathbb{R}_+^*)^n \times \mathbb{R}_+^U \times \mathbb{R}_+^{U\times n}$
- *transition labels* $\Sigma = \{s,t\} \times U \times \mathbb{R}_+^n$
- *(labelled) transition relation described below*

States *are of the form* $(R, \theta, \hat{R}) \in S$ *where:*

- $R \in (\mathbb{R}_+^*)^n$ *is the vector of reserves of the price machine*
- $\theta \in \mathbb{R}_+^U$ *is the vector of fractions of LP tokens held by users, so that* $\mathbf{1}^T\theta = 1$
- $\hat{R} \in \mathbb{R}_+^{U\times n}$ *is the vector of users' net wealths*

The net wealth of user u is defined as the amount of tokens user u would own after *withdrawing all its LP tokens. The vector $R_u^{loc} := \hat{R}_u - \theta_u R$, represents u's local reserves, that is to say the amount of capital which u has not invested as an LP.*

Labels *are of the form* $(s/t, u, \lambda)$ *where:*

- *s/t indicates whether the transition is a swap (s) or a transfer (t)*
- *u is the user causing the transition*
- *$\lambda \in \mathbb{R}_+^n$ is a vector expressing multiplicatively the change in reserves induced by the transition*

That is to say, for a transition in reserves $R \to R'$, λ is the unique (positive) vector such that $R' = \lambda R$, where multiplication is understood component-wise.

Transitions *form a ternary relation over $S \times \Sigma \times S$ and come in two types.*

Swaps: $(R, \theta, \hat{R}) \xrightarrow{s_u(\lambda)} (\lambda R, \theta, \hat{R}')$ *with* $\psi(R) = \psi(\lambda R)$.

Fractions θ of ownership are invariant under swaps (as new LP tokens are distributed proportionally). New net wealths of u and $v \neq u$ are respectively given by:

$$\begin{array}{rcl} \hat{R}'_u & = & \hat{R}_u + (1-\theta_u)(\mathbf{1}-\lambda)R \\ \hat{R}'_v & = & \hat{R}_v - \theta_v(\mathbf{1}-\lambda)R \end{array}$$

Transfers: $(R, \theta, \hat{R}) \xrightarrow{t_u(\mu)} (\mu R, \theta', \hat{R}')$

We set $\nu := \psi(\mu R)/\psi(R)$.

New ownership fractions and net wealths of u and $v \neq u$ are respectively given by:

$$\left\{\begin{array}{rcl} \theta'_u & = & \theta_u + (1-\theta_u)(1-\nu^{-1}) \\ \hat{R}'_u & = & \hat{R}_u + (1-\theta_u)(\mathbf{1}-\mu\nu^{-1})R \end{array}\right. \qquad \left\{\begin{array}{rcl} \theta'_v & = & \theta_v\nu^{-1} \\ \hat{R}'_v & = & \hat{R}_v - \theta_v(\mathbf{1}-\mu\nu^{-1})R \end{array}\right.$$

With this long definition in place, we can make a number of remarks.

As said, factors λ, μ in the (Swap) and (Transfer) transitions above are in general vectors, not scalars.

In (Swap) transitions, tokens received and paid out by the price machine can be written explicitly in multiplicative form:

$$(R' - R)_+ = (\lambda - \mathbf{1})_+ R$$
$$(R - R')_+ = (\mathbf{1} - \lambda)_+ R$$

The pre-factor $(1-\theta_u)$ occurring in u's post-swap net wealth $\hat{R}'_u$ expresses the fact that u is partly self-trading if s/he also holds LP tokens. This pre-factor can therefore be interpreted as a "wash trading" factor. In particular, if $\theta_u = 1$ (and therefore $\theta_v = 0$ for $v \neq u$) net wealths are invariant under swap; which is to be expected as, in this case, user u is entirely trading with self.

The total amount of tokens is conserved under any transition: $\mathbf{1}^T \hat{R} = \mathbf{1}^T \hat{R}'$.

Transitions are subject to a budget constraint, $\hat{R}'_u \succeq 0$ for u a user; ie user u holds non-negative amounts of tokens post-transition. For a Transfer the constraint can be written $\hat{R}'_u - \theta'_u R' \succeq 0$.

Finally, notice that $\hat{R} = \hat{R}'$ under perfectly balanced transfers (because in this case $\mu = \nu\mathbf{1}$). (This makes it convenient to work with net wealths and multiplicative transitions rather than local reserves and additive transitions.)

2.5 Single user price machines

In the following, we only need to consider the case of a single user u (the attacker) interacting with the price machine. Caveat: this does not mean that other users are not present as LPs, just that they do not interact with the machine while u is. In particular, u is allowed to have varying amounts of fraction of ownership of the pools. (A similar two-user version of a price machine is a convenient model to study the so-called MEV attacks on a price machine.)

▶ **Definition 2.** *In the single user case, the data presented in the preceding definition simplifies as follows.*

Single user states reduce to the simpler form $(R, \theta_u, \hat{R}_u) \in (\mathbb{R}_+^*)^n \times \mathbb{R}_+ \times \mathbb{R}_+^n$.

Labelled transitions simplify to:

Swaps:

$$(R, \theta_u, \hat{R}_u) \xrightarrow{s_u(\lambda)} (\lambda R, \theta_u, \hat{R}_u + (1-\theta_u)(\mathbf{1}-\lambda)R)$$

with $\psi(\lambda R) = \psi(R)$

Transfers:

$$(R, \theta_u, \hat{R}_u) \xrightarrow{t_u(\mu)} (\mu R, \theta_u + (1-\theta_u)(1-\nu^{-1}), \hat{R}_u + (1-\theta_u)(\mathbf{1}-\nu^{-1}\mu)R)$$

with $\nu := \psi(\mu R)/\psi(R)$

Let us verify that the transitions above are correct.

For a Swap $(R, \theta, \hat{R}_u) \xrightarrow{s_u(\lambda)} (\lambda R, \theta, \hat{R}'_u)$, $\theta = \theta_u$ is unchanged.

Local reserves become:

$$R'^{loc}_u := \hat{R}_u - \theta_u R - (\lambda - \mathbf{1})R$$

hence

$$\begin{aligned} \hat{R}'_u &:= R'^{loc}_u + \theta_u \lambda R \\ &= \hat{R}_u + (1-\theta_u)(\mathbf{1}-\lambda)R \end{aligned}$$

as in the expression given above.

For a Transfer $(R, \theta, \hat{R}_u) \xrightarrow{t_u(\mu)} (\mu R, \theta', \hat{R}'_u)$, the total amount of LP tokens post transition is $\psi(\mu R)$. The difference in LP tokens, $\psi(\mu R) - \psi(R)$ (negative for a withdraw), is given to u. Therefore the amount of LP tokens held by u after the transition is $\theta'_u \psi(\mu R) = \theta_u \psi(R) + \psi(\mu R) - \psi(R)$. Hence:

$$\theta'_u = \theta_u + (1-\theta_u)(1-\nu^{-1})$$

The amount given by u is $(\mu - \mathbf{1})R$, so $R'^{loc}_u = R^{loc}_u - (\mu - \mathbf{1})R$, and:

$$\begin{array}{rcl} \hat{R}'_u & = & R'^{loc}_u + \theta'_u \mu R \\ & = & R^{loc}_u - (\mu - \mathbf{1})R + (\theta_u + (1-\theta_u)(1-\nu^{-1}))\mu R \\ & = & \hat{R}_u - \theta_u R - (\mu - \mathbf{1})R + (\theta_u + (1-\theta_u)(1-\nu^{-1}))\mu R \\ & = & \hat{R}_u + (1-\theta_u)R - \mu R + (\theta_u + (1-\theta_u)(1-\nu^{-1}))\mu R \end{array}$$

and the expression given above for $\hat{R}'_u$ follows.

In the special case of a perfectly balanced transfer, $\nu^{-1}\mu = \mathbf{1}$ and $\hat{R}'_u = \hat{R}_u$, ie the net wealth of u is unchanged.

3 Consistency

Intuitively, a price machine is consistent if no attacker can extract price-independent profit, regardless of the machine's initial state. With the vocabulary developed in the preceding section, we can now formulate this precisely.

Given a price machine, a sequence of transitions caused by the same user u is called a *trace*. We say that two traces are *equivalent* if they have the same initial and final state. Clearly, equivalence is compatible with composition.

Equivalent traces may induce different budget constraints. (See an example below.)

Concretely, in a blockchain with a sequential execution model, the attacker can drive the price machine to execute any trace of his choice into one single transaction (that is to say atomically).

▶ **Definition 3** (Consistency). *A price machine is said to be consistent if, for any trace σ caused by u:*

$$(R, \theta_u, \hat{R}_u) \xrightarrow{\sigma} (R', \theta'_u, \hat{R}'_u)$$

$\hat{R}_u \preceq \hat{R}'_u$ implies $\hat{R}_u = \hat{R}'_u$.

As we will see below, it is easy to show that a trace which includes only Swap transitions cannot be a counter-example to consistency [2]. (Essentially because along such a sequence the invariant $\psi(R)$ cannot decrease.) The idea of the consistency proof is to reduce a trace into an equivalent one, where Swaps are all executed before any Transfer transition happens, and Transfers are perfectly balanced. Perfectly balanced transfers do not alter the user net wealth (as noticed a few lines above). So the conclusion would follow.

▶ **Proposition 4.** *Given a price machine with 1-homogenous trade function, an arbitrary transfer is equivalent to a swap (without fees) followed by a balanced transfer.*

Proof. We use reduced states and transitions (Def. 2). Consider an arbitrary transfer $t_u(\mu)$. Define $\nu := \psi(\mu R)/\psi(R)$.

We have the following trace:

$$\begin{array}{lll} R, \theta_u, \hat{R}_u & \xrightarrow{s_u(\nu^{-1}\mu)} & \nu^{-1}\mu R, \theta_u, \hat{R}_u + (1-\theta_u)(\mathbf{1} - \nu^{-1}\mu)R \\ & \xrightarrow{t_u(\nu\mathbf{1})} & \mu R, \theta_u + (1-\theta_u)(1-\nu_*^{-1}), \\ & & \hat{R}_u + (1-\theta_u)(\mathbf{1} - \nu^{-1}\mu)R + (1-\theta_u)(\mathbf{1} - \nu_*^{-1}\nu\mathbf{1})R \end{array}$$

with $\nu_* = \psi(\mu R)/\psi(\nu^{-1}\mu R) = \nu$, because ψ is 1-homogenous, and therefore, the above trace is equivalent to a direct Transfer $t_u(\mu)$, as can be read directly from Def. (2). We also have to check that the first transition is correct:

$$\psi(\nu^{-1}\mu R) \quad = \quad \psi(\psi(R)/\psi(\mu R)\, \mu R) \quad = \quad \psi(R)$$

again by 1-homogeneity of ψ. ◀

▶ **Proposition 5.** *Given a price machine with 1-homogenous trade function, swaps and balanced transfers due to the same user commute.*

Proof. Given a reduced state $(R, \theta_u, \hat{R}_u)$, we have

$$(R, \theta_u, \hat{R}_u) \xrightarrow{t_u(\mu)s_u(\lambda)} (\lambda\mu R, \theta_u + (1-\theta_u)(1-\nu^{-1}), \hat{R}_u + (1-\theta_u)(\mathbf{1} - \lambda\mu\nu^{-1})R),$$
$$(R, \theta_u, \hat{R}_u) \xrightarrow{s_u(\lambda)t_u(\mu)} (\lambda\mu R, \theta_u + (1-\theta_u)(1-\nu_*^{-1}), \hat{R}_u + (1-\theta_u)(\mathbf{1} - \lambda\mu\nu_*^{-1})R),$$

where $\nu = \psi(\mu R)/\psi(R) = \psi(\mu\lambda R)/\psi(\lambda R) = \nu_*$, with the middle equality because ψ is 1-homogenous. ◀

The "commutation" above has to be understood up to budget constraints. Indeed, if we consider the constant product ψ (see §2.3), with initial state $(R, \theta_u, \hat{R}_u)$, where $R = (60, 60)$, $\theta_u = 0.5$, $\hat{R}_u = (80, 80)$, and transitions $t_u(\mu)$, $s_u(\lambda)$, are defined by $\mu = (0.5, 0.5)$ (balanced), and $\lambda = (0.5, 2)$. The budget constraint prevents the expected equivalence between $t_u(\mu)s_u(\lambda)$ and $s_u(\lambda)t_u(\mu)$.

We can now wrap up the proof.

▶ **Theorem 6.** *A price machine is consistent if its trading function is* 1*-homogenous.*

Proof. Given a state $(R, \theta_u, \hat{R}_u)$ and an attack trace σ. Since the feasibility of swaps and transfers is preserved under a positive translation on $\hat{R}_u$, we may suppose that $\hat{R}_u$ is large enough (the attacker has deep pockets).

By Prop. 4 (decomposition of non-balanced transfers), σ is equivalent to a trace σ_1 where every transfer is balanced.

By Prop. 5 (postponement of transfers), σ_1 is in turn equivalent to a trace σ_2 of the form $s_u(\lambda_1)\cdots s_u(\lambda_n)t_u(\mu_1)\cdots t_u(\mu_m)$, where each $t_u(\mu_i)$ is balanced.

It is easy to see that sequences of single-user balanced transfers can be aggregated, meaning $t_u(\mu_1)\cdots t_u(\mu_m)$ is equivalent to a one step transfer $t_u(\mu)$ with $\mu = \mu_1 \cdots \mu_m$. Likewise, sequences of single-user swaps $s_u(\lambda_1)\cdots s_u(\lambda_n)$ are equivalent to a one step swap $s_u(\lambda)$ with $\lambda = \lambda_1 \cdots \lambda_n$.

By combining both remarks we can obtain a trace σ_3 equivalent to the original σ and of the simple form $s_u(\lambda)t_u(\mu)$.

Let now $\hat{R}'_u$ be the net wealth of user u at the end of σ_3 (equivalently at the end of σ). Since $t_u(\mu)$ is perfectly balanced, $\hat{R}'_u$ is also the net wealth of u after the combined swap $s_u(\lambda)$. Hence $\hat{R}'_u = \hat{R}_u + (1-\theta_u)(\mathbf{1} - \lambda)R$.

Now suppose that $\hat{R}_u \preceq \hat{R}'_u$, it must be that $\lambda \preceq \mathbf{1}$. Since $\psi(R) = \psi(\lambda R)$ (no fees) and ψ is (strictly) increasing, it must be in fact that $\lambda = \mathbf{1}$. Hence $\hat{R}_u = \hat{R}'_u$. ◀

We do not really need to suppose that swaps have zero fees. The proof above can handle the case where $s_u(\lambda_1)\cdots s_u(\lambda_n)t_u(\mu)$ consists of swaps with and without fees.

4 Applications

It remains to show that our approach applies to some interesting AMMs.

Recall that the epigraph and the hypograph of a function $f : X \to \overline{\mathbb{R}}$, where $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, are defined as:

$$\operatorname{epi}(f) := \{(x, r) \in X \times \mathbb{R} : r \geq f(x)\},$$
$$\operatorname{hyp}(f) := \{(x, r) \in X \times \mathbb{R} : r \leq f(x)\}.$$

To do this we define a specific class of candidate trading functions.

▶ **Definition 7.** *Let a triple (f, g, F) be given with:*

- f, $g : (\mathbb{R}_+^*)^n \to \overline{\mathbb{R}}_+$
- $f \leq g$
- $F : \operatorname{epi} f \cap \operatorname{hyp} g \to \mathbb{R}$

Suppose that for all $R \in (\mathbb{R}_+^*)^n$, there is a unique D such that $F(R, D) = 0$ and $f(R) \leq D \leq g(R)$. The candidate trading function associated to (f, g, F) is then defined as $\psi(R) := D$.

Intuitively, $f(R)$, $g(R)$ give bounds on the amount of LP tokens $\psi(R)$ available for a given level of reserves R.

Now we ask for sufficient conditions for such a function to be increasing, i.e. to be an actual trading function.

▶ **Theorem 8.** *Let ψ be as in Def. 7. For ψ to be increasing, it is sufficient that the following holds:*

1. *$\forall R \in \operatorname{dom} \psi$, $F(R, \cdot)$ is of class $\mathcal{C}^1$ on $[f(R), g(R)]$, with $F'(R, \cdot) < 0$*
2. *$\forall D \in \operatorname{im} \psi$, $F(\cdot, D)$ is strictly increasing*
3. *f, g are continuous, and, $f < \psi < g$ almost everywhere*

Proof. From condition 1, we know that ψ is of class $\mathcal{C}^1$, by the inverse function theorem. Since we also have condition 3, it suffices to prove that ψ is locally strictly increasing, for every R such that $f(R) < \psi(R) < g(R)$.

Now, pick $R \in \operatorname{dom} \psi$ such that $f(R) < \psi(R) < g(R)$. By condition 3 and the continuity of ψ, there is an open neighbourhood N_R of R such that for almost every $R' \in N_R$, both $f(R') < \psi(R') < g(R')$ and $f(R) < \psi(R') < g(R)$ hold.

Suppose, without loss of generality, that $R \prec R'$. By definition, we have $F(R, \psi(R)) = F(R', \psi(R')) = 0$. By condition 2, we have $F(R, \psi(R')) < F(R', \psi(R'))$, so that $F(R, \psi(R)) > F(R, \psi(R'))$. By condition 1, we have $\psi(R) < \psi(R')$. Done! ◀

▶ **Theorem 9.** *Let ψ be a trading function defined as in Def. 7 via a triple F, f, g. Suppose F is positively homogenous, and f, g are 1-homogenous. Then ψ is positively homogenous of degree 1.*

Proof. Suppose that F is positively homogenous of degree k. Pick $R \in \operatorname{dom} \psi$, and $s > 0$. We have $F(sR, s\psi(R)) = s^k F(R, \psi(R)) = 0$, therefore $F(sR, s\psi(R)) = 0$. On the other hand, by definition $\psi(R) \geq f(R)$, hence $s\psi(R) \geq sf(R) = f(sR)$. Similarly $\psi(R) \leq g(R)$, hence $s\psi(R) \leq sg(R) = g(sR)$ Therefore by unicity of the solution to $F(sR, \_) = 0$ in the interval $[f(sR), g(sR)]$, it must be that $s\psi(R) = \psi(sR)$. ◀

4.1 The Balancer family

Balancer v1 is a multi-token AMM that generalizes [7] Uniswap v2's [1]. It is one of the formulas supported by Balancer v2. The trading function for Balancer with n tokens, denoted by ψ_B, is defined as in the general framework, with $f := 0$, $g := +\infty$, and

$$F(R, D) := \prod_{i=1}^{n} R_i^{w_i} - D,$$

where the weights in $w \in (\mathbb{R}_+^*)^n$ satisfy $\mathbf{1}^T w = 1$.

The trading function for Uniswap v2 (seen above) can be seen as a special case of that of Balancer, with $n = 2$ and $w_1 = w_2 = 0.5$.

▶ **Proposition 10.** *ψ_B is increasing and positively homogenous of degree 1.*

Proof. Clear. We even have $\psi_B(R) := \prod_{i=1}^n R_i^{w_i}$. ◀

▶ **Corollary 11.** *The Balancer family is consistent.*

We can remark that the on-chain implementation of a non-balanced deposit $R \to^t R'$ actually only returns $\tau := \min_i(R' - R)$ LP tokens [9], so that the balanced deposit $R \to \tau R$ is already more advantageous. A *reasonable* user always makes balanced transfers.

4.2 Application: the Curve family

Curve v1 is a multi-token AMM that offers arguably fairer trades than Uniswap v2 [5]. It is also one of the formulas supported by Balancer v2. Curve v2 is based on the same idea as Curve v1, but with some notable changes [6]. In these AMMs, transfers do not need to be balanced.

The trading function(s) for Curve with n tokens, denoted by ψ_C, is defined as in the general framework:

- $f(R) := n(\prod_{i=1}^n R_i)^{\frac{1}{n}} \leq \mathbf{1}^T R =: g(R)$
- with

$$F(R,D) \quad := \quad D^n \left[K(R,D)\left(g(R)D^{-1} - 1\right) + n^{-n}\left(\left(f(R)D^{-1}\right)^n - 1\right)\right]$$

where $K(R,D)$ is defined as either of:

$$\begin{array}{rcll} K(R,D) & := & A(f(R)D^{-1})^n & \textit{(Curve v1)} \\ & := & A(f(R)D^{-1})^n \frac{\gamma^2}{(\gamma+1-(f(R)D^{-1})^n)^2} & \textit{(Curve v2)} \end{array}$$

and $A \geq 0$ is called the *amplification coefficient*, and $\gamma > 0$ is a small constant.

▶ **Proposition 12.** *F is positively homogenous of degree n.*

Proof. Clear, since f, g are 1-homogenous, and K is 0-homogenous. ◀

▶ **Proposition 13.** *Condition 1 is verified.*

Proof. It's not hard to check that $\forall D \in \pi_{n+1} \circ \operatorname{dom} F$, for Curve v1,

$$F'(R,\cdot)(D) \quad = \quad -D^{n-1}n^{-n+1} - D^{-2}Af(R)^n g(R) \quad < \quad 0$$

and, for Curve v2, by denoting $T(R,D) := \gamma + 1 - (f(R)D^{-1})^n > 0$,

$$\begin{array}{rcll} F'(R,\cdot)(D) & = & -D^{n-1}n^{-n+1} - D^{-2}(f(R))^n g(R) A\gamma^2 T(R,D)^{-2} & \\ & & -2D^{-n-2}(f(R))^{2n}(g(R)-D)An\gamma^2 T(R,D)^{-3} & \\ & < & -D^{n-1}n^{-n+1} & < \quad 0, \end{array}$$

so that condition 1 is verified. ◀

▶ **Proposition 14.** *ψ_C is well-defined. In addition, condition 3 is verified.*

Proof. We have $\forall R \in (\mathbb{R}_+^*)^n$, $F(R, g(R)) \leq 0 \leq F(R, f(R))$, so that ψ_C is well-defined, by the arithmetic-geometric mean inequality. Since the equalities hold if and only if $R_1^{-1}R = \mathbf{1}$, condition 3 is verified. ◀

It is worth noting that for Curve v1, F can be analytically continued to $(\mathbb{R}_+^*)^n \times \mathbb{R}_+^*$, where $\forall R \in (\mathbb{R}_+^*)^n$, $F(R, 0_+) > 0$, and $F'(R,\cdot)$ has at most one stationary point, so that $\psi_C(R)$ is in fact the unique solution of $F(R,\cdot) = 0$ on $\mathbb{R}_+^*$.

▶ **Proposition 15.** *Condition 2 is verified.*

Proof. Clearly, it suffices to prove that $\forall D \in \operatorname{im}\psi$, $K(\cdot, D)$ is strictly increasing. This is clear for Curve v1, but also for Curve v2, by the fact that $(fD^{-1})^n \leq 1 \leq \gamma + 1$, and that $(fD^{-1})^n$ is strictly increasing on $\operatorname{dom} K(\cdot, D)$. ◀

▶ **Corollary 16.** *ψ_C is strictly increasing and homogenous of degree 1.*

▶ **Corollary 17.** *The Curve family is consistent.*

5 Conclusion

We have proved the consistency of a specific family of (zero-fees) AMMs which are based on increasing and 1-homogenous invariants (also called trading functions). This is a new result. Note again that consistency says nothing of an AMM's suitability or efficiency as a market mechanism, just that it is not outright flawed.

Some generalisations can be expected. For example, while it is fairly intuitive that the consistency of AMMs with fees follows from that without, it remains to be proven rigorously.

One way to use our result is in the course of designing new AMMs. For instance, it follows directly from our result that the trading function $\psi(X, Y) = (X^3Y + XY^3)^{1/4}$ which has been proposed recently generates a consistent AMM. In the same vein, one could generalise the Curve approach of mixing two existing price machines (in the specific Curve construction one mixes the linear $X + Y$ invariant and the product XY one) to obtain a general consistency-preserving mixing combinator on the space of price machines.

However, note that our method only offers a sufficient condition. A typical example that does not fall under our result is the non-homogenous trading function $\psi(X, Y) = X+Y+XY$. It can be shown independently that this particular choice is indeed inconsistent in the sense of Def. 3 and can be exploited, but the results obtained in this paper do not give us a specific method to look for such an attack.

The reader might wonder if our approach applies also to the Uniswap v3 protocol. Uniswap v3 is not exactly an AMM but rather a (n efficient) aggregator of AMMs each based on a concentrated version of the original (Uniswap v2) product invariant. In general, in the case of protocols aggregating AMMs, the consistency question boils down to the consistency of the individual AMMs being aggregated. Now in Uniswap v3 every LP-position has a single liquidity provider, hence transfers are trivial ($\theta_u = 1$ at all times in the language of Section 2) and consistency follows from the monotony of the (concentrated) product invariant.

References

1. Hayden Adams, Noah Zinsmeister, and Dan Robinson. *Uniswap v2 Core*, 2020.
2. Guillermo Angeris and Tarun Chitra. Improved price oracles. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. ACM, 2020. doi:10.1145/3419614.3423251.
3. CoinMarketCap. Top cryptocurrency decentralized exchanges ranked, 2022. URL: https://coinmarketcap.com/rankings/exchanges/dex/.
4. CryptoSec. Comprehensive list of defi hacks and exploits, 2022. URL: https://cryptosec.info/defi-hacks/.
5. Michael Egorov. Stableswap-efficient mechanism for stablecoin liquidity. *Retrieved Feb*, 24:2021, 2019.
6. Michael Egorov. Automatic market-making with dynamic peg. *Retrieved Dec 2021*, June 2021. URL: https://curve.fi/files/crypto-pools-paper.pdf.

7 Fernando Martinelli and Nikolai Mushegian. *Balancer: A non-custodial portfolio manager, liquidity provider, and price sensor*, 2019.

8 Nick Szabo. Formalizing and securing relationships on public networks, September 1997. URL: https://firstmonday.org/ojs/index.php/fm/article/view/548.

9 Uniswap. Core smart contracts of Uniswap v2, 2020. URL: https://github.com/Uniswap/v2-core/blob/master/contracts/UniswapV2Pair.sol.

10 Chiqing Zhang. Test cases implementing the example, 2022. retrieved Sep 2022. URL: https://github.com/zhangchiqing/v2-periphery/blob/master/test/WalkThrough.spec.ts.