Amber Whitepaper

André Gaul Aaron Lindner Andy Wermke

July 5, 2022

Abstract

In this paper we present a novel design for automated market making which separates the concerns of providing token liquidity vs. taking on certain risks. This new design allows for low-risk single-sided liquidity provision and thus enables Amber to tap new sources of liquidity and operate with significantly lower slippage and fees compared to other AMM designs.

1 Introduction

Automated Market Makers (AMMs) have been one of the most impactful innovations of DeFi. They not only allow for fully automated decentralized trading, but as well democratize the opportunity to earn yield from market making and trading fees.

Providing liquidity to an AMM like Uniswap [1, 2] can be a very profitable activity, but it comes with a set of inherent risks. Liquidity providers (LPs) provide liquidity to a 2-sided[1] pool that always takes on the other side in the trades users make - which results in changes in the amount of tokens in the pool and thus the price they are traded for. This leads to two risks for the LP to take on[2]:

- The market risk: the market prices of the tokens in a pool may go up or down. As LPs own a share of the whole pool, they are always exposed to the market rates of two (or even more) different tokens.
- The impermanent loss (IL) risk: due to the passive price finding mechanism of classic AMM designs, each relative change in the pool token prices results in a relative loss compared to the scenario of just holding the pool tokens.

Whilst several newer DeFi protocols like Ondo [5], Tokemak [6] or Bancor v3 [7] allow IL-free single-sided liquidity provision (LPing), they're all build around classic AMM designs and thus can only take over or compensate the occurring IL, but not avoid it in the first place. Dodo [8] introduced the concept of a *Proactive Market Maker* (PMM) which can in theory eliminate the IL by setting the price actively via an external oracle, instead of finding it passively via arbitrage. Assuming a perfect oracle (which means no loss to arbitrage),

[1] There are as well notable AMMs with pools that consist of more than 2 tokens (e.g. Balancer [3] or Curve [4]), but such a design has no implications on our points of discussion.

[2] Not considering black swan risks like smart contract exploits.

the IL is then replaced by what we call *inventory risk*: Whilst there is no IL anymore, the LP is still exposed to the risk that comes from diverging pool token prices combined with fluctuating relative amounts of those tokens in the pool[3]. As the PMM effectively takes on the other side of each user trade at the "fair" external market rate, the profit/loss of this inventory risk fluctuates around zero (in contrast to IL always being negative) - but the relative loss (or profit) at a given point in time can still be significant.

In addition, the market risk prevails for PMMs. Even if most PMMs allow single-sided deposits of only one of the pool tokens, the LP in exchange still receives a share of the whole pool, and is thus exposed to the market risk of all tokens in that pool.

PMMs can be a big step forward in terms of capital efficiency, risk reduction and revenue optimization for assets where highly reliable and manipulation-resistant oracle price feeds exist, i.e. in cases where the price finding is clearly dominated by off-chain markets. But significant inventory and market risks remain, and thus still prevent low-risk capital from LPing. This is especially true when looking at the Forex use case: A low-risk option for single-sided LPing would allow attracting deep liquidity in different fiat-pegged stable tokens, as yields on off-chain savings accounts are very low in most major economies. And deep liquidity in different fiat-pegged stable tokens is the precondition for making *Decentralized Forex* (DeFo) a viable option.

The proposed Amber PMM solves this challenge through separation of concerns: We disentangle the service of providing liquidity from the service of taking on certain risks, and thus allow for the first time to provide liquidity without IL-risk, inventory risk or market risk for any token other than the one provided.

To achieve this, we use single-sided swap pools and introduce an incentivized "backstop pool" which takes on the inventory risk, and which rather targets seasoned DeFi users as LPs. Thus the normal Amber swap pools offer a very low risk profile for single-sided LPs, and can target retail users and traditional finance.

This separation of concerns is complemented by custom price curves for each swap pool which allow for an optimal liquidity concentration around the oracle price for every token, and elaborated pool rebalancing mechanisms. Altogether this setup allows for extremely low slippage and trading fees.

2 Amber's Design

Amber introduces a novel PMM design, which consists of a number of swap pools, each holding an asset[4] that can be traded on Amber, and one backstop pool holding a stablecoin reserve. A price oracle is required for each asset that is to be traded.

Every pool comes with a coverage ratio, as recently introduced by Platypus [9], that states how much funds are currently held by the pool compared to

[3]Example: due to trades a pool has more of token A and less of token B compared to it's equilibrium - and later the price of token A decreases significantly, whilst token B doesn't change in price.

[4]throughout this paper we will use the more general term *asset* instead of *token* for formal definitions of the Amber design - unless we refer to a specific token or an actual smart contract implementation

how much is owed to liquidity providers. Each pool is initially in an equilibrium. Trades and certain other actions that drive pools away from their equilibrium are penalized with a negative slippage, whereas certain trades and other actions that move the coverage closer to the equilibrium are incentivized with a positive slippage.

The backstop pool is used when a liquidity provider seeks to withdraw funds from a swap pool with an already low coverage ratio. In this case liquidity from the backstop pool can be used for the withdrawal. The backstop pool therefore takes on most of the system risks and in return receives a substantial share of the $AMBR token incentives.

In this section we provide a mathematical framework of AMMs and introduce the Amber AMM.

2.1 Notation

Throughout the paper e_i denotes the i-th standard unit (column) vector and we omit stating its dimension when it is clear from where it is used. That means that e_i denotes a vector consisting of zeros only, except for a single one at index i. Transposing is denoted by T, e.g., e_i^T for a unit row vector. Note that i can be zero-based.

2.2 Definitions

In the following we assume that liquidity providers can provide liquidity for a reserve asset X_0 (backstop pool) and $n \in \mathbb{N}$ different assets $X_1, \ldots, X_n$ (swap pools).

Definition 2.1 (Asset Space)**.** For an AMM with n assets we define the *asset space* $V_n = \mathbb{R}_{\geq 0}^{n+1}$ of non-negative numbers. For $[x_0, x_1, \ldots, x_n]^T \in V_n$, x_0 is the amount of the reserve asset X_0 and x_i is the amount of asset X_i for $i \in \{1, \ldots, n\}$.

Definition 2.2 (State)**.** For an AMM with n assets and m liquidity providers, we define the AMM's *state* as a tuple $s = (b, L)$ where

1. $b = [b_0, \ldots, b_n]^T \in V_n$ is the *balance vector* and
2. $L = [L_1, \ldots, L_m] \in \mathbb{R}_{\geq 0}^{(n+1)\times m}$ is the *liability matrix* where $L_j \in V_n$ is the AMM's liability for liquidity provider $j \in \{1, \ldots, m\}$.

The *state space* is thus defined by $S_{n,m} := V_n \times \mathbb{R}_{\geq 0}^{(n+1)\times m}$.

Remark 2.3. The balance vector contains the asset amounts that the AMM currently holds and the liability matrix captures the asset amounts that the AMM owes to liquidity providers. Deposits by liquidity providers increase the balance and liability. Trades only affect the balance vector. In the case of an imbalance of the balance vector and the liabilities, withdrawals can be realized with assets that differ from the ones that have originally been deposited. Precise definitions for all operations of the AMM will be given in the course of this paper.

In practice, the liabilities are recorded by issuing tokens to liquidity providers for deposits and by consuming tokens for withdrawals. The tokens can change ownership so the columns in the liability matrix can be "dynamic". However, the sum of the columns does not change by ownership changes.

Definition 2.4 (Price)**.** $p = [p_0, p_1, \ldots, p_n]^T \in \mathbb{R}_+^{n+1}$ is called a *price vector* where, for $i \in \{0, \ldots, n\}$, p_i is the price of asset X_i in terms of a reference asset R.

Remark 2.5. In practice, the price vector p will carry data from *external markets* that is pulled from a price oracle. Note that if the reserve asset X_0 is a stablecoin (e.g., \$USDC) and the reference asset is the underlying fiat asset (e.g., USD), then $p_0 = 1$ does not necessarily hold and another price conversion from X_0 to that fiat asset R is required.

In order to compare amounts of different assets, we convert the amounts into amounts of the reference asset R:

Definition 2.6. Given a price vector p, we define the *R-conversion function* $\pi : R^{n+1} \to R^{n+1}$ by

$$\pi(v) := [v_0 p_0, \ldots, v_n p_n]^T \text{ for a } v = [v_0, v_1, \ldots, v_n]^T \in R^{n+1}.$$

Definition 2.7. For a state $s = (b, L) \in S_{n,m}$ we define:

1. The *liability* as $l(s) = [l_0(s), \ldots, l_n(s)]^T := \sum_{j=1}^m L_j \in V_n$.

2. The *coverage ratio* $\theta : S_{n,m} \to \big(\mathbb{R}_{\geq 0} \cup \{\infty\}\big)^{n+1}$ by

$$\begin{aligned} \theta(s) :=& [\theta_0(s), \ldots, \theta_n(s)]^T, \\ \text{where } \theta_i(s) =& \begin{cases} \frac{b_i}{l_i(s)} & \text{if } l_i(s) > 0 \\ 1 & \text{if } l_s(s) = b_i = 0 \\ \infty & \text{otherwise} \end{cases} \qquad \text{for } i \in \{0, \ldots, n\}. \end{aligned}$$

3. The *liquidity gap* $\gamma : S_{n,m} \to \mathbb{R}^{n+1}$ by

$$\gamma(s) = [\gamma_0(s), \ldots, \gamma_n(s)]^T := b - l(s).$$

Remark 2.8. In practice we can avoid the case of zero liabilities and thus $\theta_i(s) = \infty$ by making small deposits to each pool that will never be withdrawn.

With the help of the liquidity gap and the R-conversion function, we can observe some useful conditions for a state $s \in S_{n,m}$ of the AMM. Let $\overline{\gamma} = [\overline{\gamma}_0, \ldots, \overline{\gamma}_n]^T := \pi\big(\gamma(s)\big)$ be the R-converted liquidity gap.

1. If $0 \leq \gamma(s)$, then all liabilities can be paid to the liquidity providers in the assets they originally deposited.

2. If $\sum_{i=1}^n \max\{-\overline{\gamma}_i, 0\} \leq \overline{\gamma}_0$, then all liabilities can be paid out with the originally deposited assets or via the reserve asset X_0.

3. If $\sum_{i=0}^n \overline{\gamma}_i \geq 0$, then all liabilities can be paid out to the liquidity providers, but potentially in other assets than the originally deposited asset or the reserve asset X_0.

The above conditions help to determine the "health" of the AMM.

2.3 Actions

The AMM can be interacted with by applying certain *actions*.

Definition 2.9. Let us define the following actions:

1. $\sigma_D(\Delta, i, j)$: a *deposit* of the amount $\Delta \in \mathbb{R}_+$ of asset X_i for an $i \in \{0, \ldots, n\}$ by liquidity provider $j \in \{1, \ldots, m\}$. The deposit increases the balance for asset X_i by Δ and the liability for provider j and asset X_i by $\Delta + \alpha$ where $\alpha \in \mathbb{R}_{\geq 0}$ is a potential reward.

2. $\sigma_W(\Delta, i, j)$: a *withdrawal* of the amount $\Delta \in \mathbb{R}_+$ of asset X_i for an $i \in \{0, \ldots, n\}$ by liquidity provider $j \in \{1, \ldots, m\}$. The withdrawal requires $\Delta \leq e_i^T L e_j$ and decreases the liability of asset X_i for provider j by Δ and the balance for asset X_i by $\Delta - \alpha$ where $\alpha \in [0, \Delta]$ is a potential penalty.

3. $\sigma_{W_R}(\Delta, i, j)$: a *reserve withdrawal* of the amount $\Delta \in \mathbb{R}_+$ of asset X_i for an $i \in \{0, \ldots, n\}$ by liquidity provider $j \in \{1, \ldots, m\}$. The reserve withdrawal requires $\Delta \leq e_i^T L e_j$ and decreases the liability for provider j by Δ but instead of a payout in asset X_i, the amount Δ is first converted to the reserve asset X_0 amount $\Delta_R = \frac{p_i}{p_0}\Delta$. Then the amount Δ_R is deducted from the asset balance for asset X_0

4. $\sigma_T(\Delta_i, i, j)$: a *trade* of the amount $\Delta_i \in \mathbb{R}_+$ of asset X_i (for $i \in \{1, \ldots, n\}$) into an asset X_j (for $j \in \{1, \ldots, n\} \setminus \{i\}$). For calculating the amount $\Delta_j \in \mathbb{R}_+$ of asset X_j that the trader receives, the amount Δ_i is first converted to $\underline{\Delta}_j = \frac{p_i}{p_j}\Delta_i$. Then $\Delta_j = \underline{\Delta}_j - \alpha$ is used where $\alpha \in [0, \underline{\Delta}_j]$. Again, the amount α can be seen as a penalty(or reward) and still needs to be defined. The trade requires $\Delta_j \leq e_j^T b$ which is deducted from the asset balance for asset X_j while the asset balance for asset X_i is increased by Δ_i.

We define Σ as the set of all actions.

Note that the above actions merely describe *intended* actions. An intended action can be rejected if it cannot be applied to the current state. The precise definition of how the state changes by applying the above actions will be given in the state transitions in section 2.5.

2.4 Slippage Function

The aim of the AMM is to facilitate trades close to the price at external markets and to keep the coverage ratios of all assets at 1 (or slightly above). For certain actions that drive the coverage ratio of a pool away from 1, we deduct an amount from the receivables for the agent which we call *slippage*. Similarly, we give rewards for certain actions that drive the coverage ratio of a pool closer to 1. In the following, we focus on the penalty case but we will use the same functions for constructing appropriate rewards in Section 2.5.

In the Platypus AMM [9], a *slippage function* was introduced to define how much is deducted. We will use a similar approach here and introduce improved and simplified slippage functions.

Let us assume we have a slippage function $h : S_{n,m} \to [0,1]^{n+1}$ where $h_i(s)$ denotes the (infinitesimal) fraction of how much is deducted at a given state $s = (b, L) \in S_{n,m}$.

To demonstrate the purpose of the slippage function, we now assume a liquidity provider $j \in \{1, \ldots, m\}$ wishes to withdraw an amount of $\Delta \leq \min\{b_i, e_i^T L e_j\}$ of asset X_i for $i \in \{1, \ldots, n\}$. While the total liabilities for X_i are decreased from $l_i(s)$ to $l_i(s) - \Delta$, the asset balance is decreased from b_i to $b_i - \Delta + \alpha$ for some $\alpha \in [0, \Delta]$. For determining α we integrate the slippage function for all intermediate states that result from the unaltered action:

$$\alpha = \int_0^\Delta h(b - te_i, L - te_i e_j^T)dt. \tag{1}$$

A crucial measure is the coverage ratio $\theta_i(s)$ of an asset X_i and thus here we restrict ourselves to slippage functions that are based on the coverage ratio, i.e., $h = g \circ \theta_i$ for a function $g : \mathbb{R}_{\geq 0} \to [0,1]$. Note that Platypus also uses a coverage-ratio-based slippage function but calculates resulting amounts with a difference quotient

$$\alpha = \frac{g(r') - g(r)}{r' - r}\Delta. \tag{2}$$

where the slippage function is only evaluated at the initial coverage ratio r and the resulting coverage ratio r'. Note that we omitted price conversions and additional swap fees for the sake of clarity. In contrast to Platypus, we achieve more accurate results by calculating a slippage function integral as in Equation (1) instead of approximating it with a difference quotient as in Equation (2).

We now investigate candidates for g which should fulfill the the following requirements:

1. g is continuous,
2. $g(0) = 1$,
3. $g(1) = 0$, and
4. $\lim_{r\to\infty} g(r) = \beta \in [0,1]$.

Property 1 is helpful for the analysis and keeps the AMM's behavior predictable for traders and liquidity providers. Together with property 2 we achieve high penalties or rewards when a pool is drained while property 3 results in low penalties or rewards around a balanced pool. Property 4 defines how a pool is treated where the balance is much greater than the liabilities.

Example 2.10. A simple choice is the rational function

$$g(x) = 1 - \left(\frac{2x}{1+x^2}\right)^{\frac{1}{k}}$$

with a parameter $k > 0$ that controls the curvature. Figure 1 shows the function for different values of k.

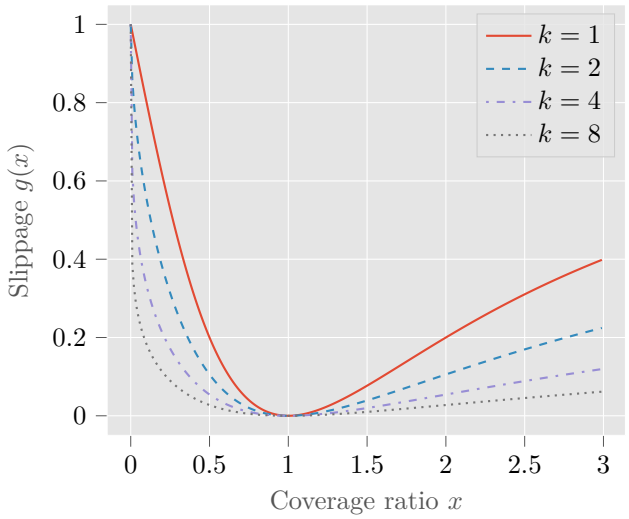

Figure 1: Slippage function g from Example 2.10 for different values of k.

Example 2.11. Another choice is the rational function

$$g(x) = \left(\frac{1 - x}{1 + \beta^{-\frac{1}{2k}} x} \right)^{2k}$$

with a parameter $k \in \mathbb{N}$ that controls the curvature and a parameter $\beta \in [0, 1]$ that defines the asymptotic value for $x \to \infty$. Figures 2 and 3 show the function for different values of k and β.

Example 2.12. Let us consider the function

$$g(x) = (1 - xe^{1-x})^k$$

with a parameter $k > 0$ that controls the curvature. Figure 4 shows the function for different values of k. The integral of g can be computed easily for $k \in \mathbb{N}$. For example, for $k = 1$ the integral is given by

$$\int g(x)dx = x + (x + 1)e^{1-x} + C$$

with a constant $C \in \mathbb{R}$. For higher values of k the integrals can still be solved analytically, e.g., with SymPy [10], but since the expressions get more complex we only present the case $k = 1$ here.

Note that the above slippage functions show a different behavior than the piecewise-defined function in Platypus. Firstly, the above functions use closed-form expressions which simplifies the presentation and analysis. Secondly, the slippage functions give truly zero slippage at coverage ratio 1 which is in contrast to Platypus which gives a slippage of some small $k > 0$ ($k = 2 \cdot 10^{-5}$ was suggested in [9]).

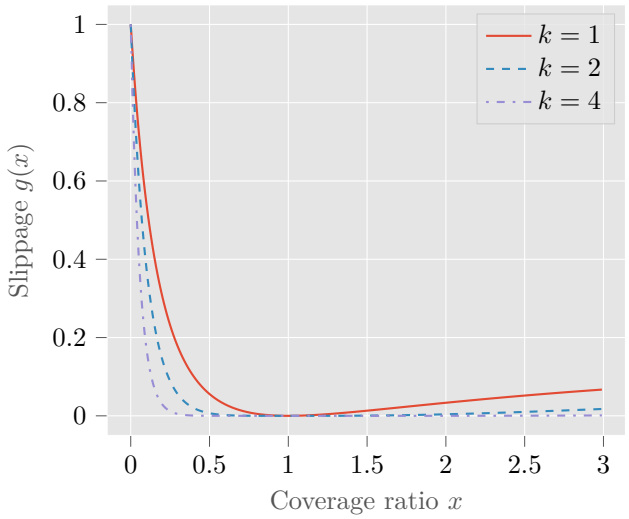

Figure 2: Slippage function g from Example 2.11 for $\beta = 0.5$ and different values of k.

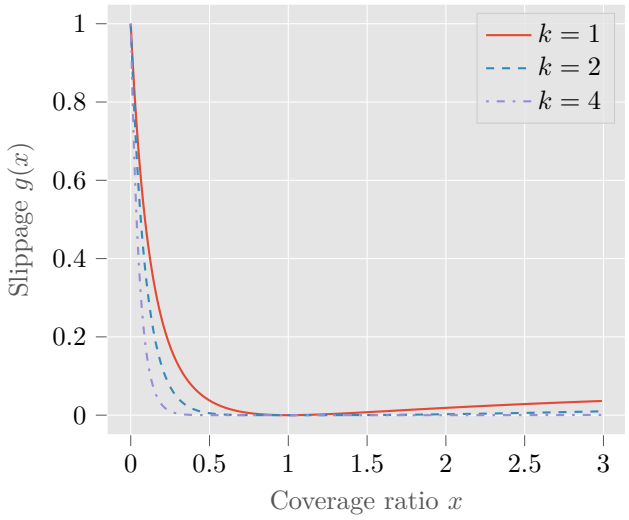

Figure 3: Slippage function g from Example 2.11 for $\beta = 0.1$ and different values of k.

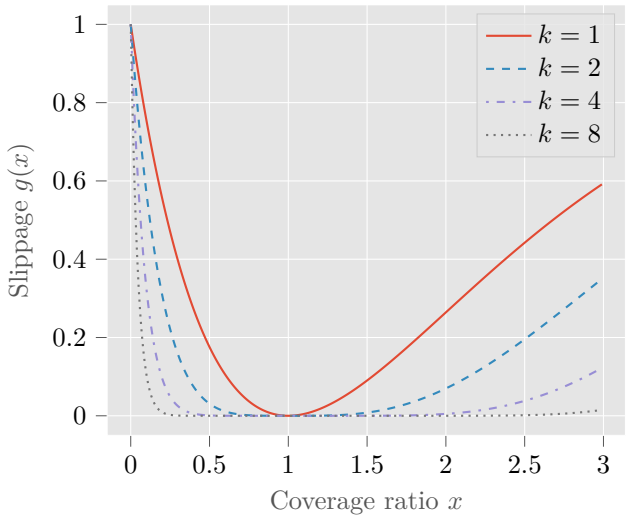

Figure 4: Slippage function g from Example 2.12 for different values of k.

The presented slippage functions (and potentially additional candidates) deserve a thorough performance analysis and comparison which is beyond the scope of this paper.

2.5 State Transitions

In this section we precisely define how actions take the AMM from one state to the next by applying a *state-transition function* $\delta : S_{n,m} \times \Sigma \to S_{n,m}$. We assume the current state is $s = (b, L) \in S_{n,m}$ and for each action $\sigma \in \Sigma$ we need to determine the next state $\delta(s, \sigma) = s' = (b', L') \in S_{n,m}$ or that the action is rejected (in which case we formally set $s' = s$).

2.5.1 Swap pool deposit

LPs can deposit single-sided liquidity into swap pools. Such a deposit alters both the liabilities and the balance of the respective pool, and thus always moves the coverage ratio $\theta_i(s)$ closer to 1 (unless $\theta_i(s) = 1$). In case $\theta_i(s) < 1$, the deposit is therefore rewarded via a positive slippage ("reward") α_D.

For a deposit $\sigma_D(\Delta, i, j)$ of the amount Δ of asset X_i by liquidity provider j, the next state is thus given by $b' = b + \Delta e_i$ and $L' = L + \Delta e_i e_j^T + \alpha_D$ where

$$\alpha_D = \begin{cases} 0 & \text{if } \theta_i(s) \geq 1 \\ \int_0^\Delta h(b + te_i, L + te_i e_j^T)dt & \text{otherwise} \end{cases} \tag{3}$$

is the deposit reward. The liquidity provider is thus entitled to withdraw $\Delta + \alpha_D$ for the deposit of Δ. Note that a withdrawal incurs a respective penalty as long as $\theta_i(s) < 1$, see Section 2.5.2.

For the slippage function g in Example 2.12 with $k = 1$, the integral in Equation (3) can be calculated with $h = g \circ \theta_i$ by

$$\begin{aligned}\int_0^{\Delta} h(b + te_i, L + te_i e_j^T)dt &= \int_0^{\Delta} g\left(\frac{b_i + t}{l_i + t}\right)dt \\ &= \int_0^{\Delta} 1 - \frac{b_i + t}{l_i + t} e^{1 - \frac{b_i + t}{l_i + t}} dt \\ &= \left[t - (l_i + t)e^{1 - \frac{b_i + t}{l_i + t}}\right]_0^{\Delta}\end{aligned}$$

where $l_i = l_i(s)$.

2.5.2 Swap pool withdrawal

LPs can withdraw their single-sided liquidity from swap pools. Such a withdrawal decreases both the liabilities and the balance of the respective pool, and thus always move the coverage ratio $\theta_i(s)$ further away from 1 (unless $\theta_i(s) = 1$). In case $\theta_i(s) < 1$, the withdrawal is therefore penalized with a negative slippage ("penalty"). The LP can choose to avoid the penalty by withdrawing from the backstop pool instead (see next section).

For a withdrawal $\sigma_W(\Delta, i, j)$ of the amount Δ of asset X_i by liquidity provider j, we first calculate the withdrawal penalty

$$\alpha_W = \begin{cases} 0 & \text{if } \theta_i(s) \geq 1 \\ \int_0^{\Delta} h(b - te_i, L - te_i e_j^T)dt & \text{otherwise.} \end{cases} \tag{4}$$

The withdrawal needs to be rejected if one of the following conditions holds:

1. If $\Delta > e_i^T L e_j$ then the withdrawal exceeds the deposited amount of liquidity provider j and needs to be rejected.

2. If $\Delta - \alpha_W > e_i^T b$ then the withdrawal exceeds the asset balance and needs to be rejected.

If it is not rejected, the liquidity provider receives the amount $\Delta - \alpha_W$ and the next state is thus given by $b' = b - \Delta e_i + \alpha_W$ and $L' = L - \Delta e_i e_j^T$.

For the slippage function g in Example 2.12 with $k = 1$, the integral in Equation (4) can be calculated with $h = g \circ \theta_i$ by

$$\begin{aligned}\int_0^{\Delta} h(b - te_i, L - te_i e_j^T)dt &= \int_0^{\Delta} g\left(\frac{b_i - t}{l_i - t}\right)dt \\ &= \int_0^{\Delta} 1 - \frac{b_i - t}{l_i - t} e^{1 - \frac{b_i - t}{l_i - t}} dt \\ &= \left[t + (l_i - t)e^{1 - \frac{b_i - t}{l_i - t}}\right]_0^{\Delta}\end{aligned}$$

where $l_i = l_i(s)$.

2.5.3 Swap pool reserve withdrawal

In case an LP wants to withdraw liquidity from a swap pool X_i with $\theta_i(s) < 1$ for $i \in \{1, \dots, n\}$, there are two options: they can either accept the withdrawal penalty when withdrawing the provided asset X_i as outlined in Section 2.5.2, or they can withdraw the full amount without penalty in asset X_0 from the backstop pool instead. Such a reserve withdrawal changes only the liability of the swap pool, but not the balance, and thus increases the coverage ratio of the swap pool (whilst decreasing the coverage ratio of the backstop pool). A reserve withdrawal is only possible up to the point where the coverage ratio of asset X_i reaches 1.

For a reserve withdrawal $\sigma_{W_R}(\Delta_i, i, j)$ of the amount Δ_i of asset X_i by liquidity provider j, we first define the maximal amount $\underline{\Delta}_i$ that can be withdrawn via the reserve pool by

$$\underline{\Delta}_i := \max\left\{x \in [0, \Delta_i] \;\middle|\; \theta_i(b, L - xe_ie_j^T) \leq 1\right\}.$$

Then we calculate the corresponding amount in the reserve asset by $\Delta_0 := \frac{p_i}{p_0}\underline{\Delta}_i$. The withdrawal needs to be rejected if one of the following conditions holds:

1. If $\Delta_i > e_i^T L e_j$ then the withdrawal exceeds the deposited amount of liquidity provider j and needs to be rejected.
2. If $\Delta_0 > e_0^T b$ then the withdrawal amount exceeds the balance of the backstop pool and needs to be rejected[5].
3. If $\Delta_i - \underline{\Delta}_i > e_i^T b$ then the withdrawal's amount of asset X_i exceeds the asset balance and needs to be rejected.

If it is not rejected the liquidity provider receives Δ_0 of the reserve asset X_0 and $\Delta_i - \underline{\Delta}_i$ of asset X_i. The next state is thus given by $b' = b - \Delta_0 e_0 - (\Delta_i - \underline{\Delta}_i)e_i$ and $L' = L - \Delta_i e_i e_j^T$.

Note that the withdrawal $\Delta_i - \underline{\Delta}_i$ for asset X_i (if any) is carried out without any penalty because the coverage ratio satisfies $\theta_i(s') = 1$ if $\Delta_i - \underline{\Delta}_i > 0$.

2.5.4 Swap

A trade $\sigma_T(\Delta_i, i, j)$ of the amount Δ_i of asset X_i to asset X_j consists of two steps.

First, because the balance of asset X_i is increased, the coverage ratio of X_i is increased and thus the trader may benefit from a reward for bringing the coverage ratio closer to 1 if it was lower than 1, or incur a penalty for worsening the coverage ratio if it is greater than 1. We split the swap-in of Δ into two parts by defining

$$\Delta_i' := \min\left\{\max\{0, l_i(s) - b_i\}, \Delta_i\right\} \in [0, \Delta].$$

The amount that is actually going to be converted is then defined by $\underline{\Delta}_i = \Delta_i + \alpha_i$ where α_i is a reward or penalty that is given by

$$\alpha_i = \int_0^{\Delta_i'} h(b + te_i, L)dt - \int_{\Delta_i'}^{\Delta_i} h(b + te_i, L)dt. \tag{5}$$

[5] See Section 4.3 for details.

Thus, a reward is generated in the interval $[0, \Delta_i']$ and a penalty is generated in the interval $[\Delta_i', \Delta_i]$. Note that the intervals can collapse to a single point in which case the corresponding interval is zero.

Secondly, the converted amount is determined by $\Delta_j := \frac{p_i}{p_j}\underline{\Delta}_i$. If $\Delta_j > e_j^T b$ then the swap needs to be rejected. Otherwise, we proceed analogously to the swap-in and split the swap-out into two parts by defining

$$\Delta_j' := \min\left\{ \max\{0, b - l\}, \Delta_j \right\} \in [0, \Delta_j].$$

The trader then receives $\underline{\Delta}_j = \Delta_j - \alpha_j$ where

$$\alpha_j = \int_{\Delta_j'}^{\Delta_j} h(b - te_j, L)dt \tag{6}$$

is a penalty for the interval $[\Delta_j', \Delta_j]$ because the balance reduction of asset X_j also reduces its coverage ratio below 1 in this interval. In the interval $[0, \Delta_j']$ no penalty or reward is applied. The next state is given by $b' = b + \Delta_i e_i - \underline{\Delta}_j e_j$ and $L' = L$.

For the slippage function g in Example 2.12 with $k = 1$, the integral in Equation (5) can be calculated with $h = g \circ \theta_i$ by

$$\begin{aligned}
\alpha_i &= \int_0^{\Delta_i'} g\left(\frac{b_i + t}{l_i}\right)dt - \int_{\Delta_i'}^{\Delta_i} g\left(\frac{b_i + t}{l_i}\right)dt \\
&= \int_0^{\Delta_i'} 1 - \frac{b_i + t}{l_i} e^{1 - \frac{b_i + t}{l_i}} dt - \int_{\Delta_i'}^{\Delta_i} 1 - \frac{b_i + t}{l_i} e^{1 - \frac{b_i + t}{l_i}} dt \\
&= \left[t + (b_j + l_i + t)e^{1 - \frac{b_i + t}{l_i}}\right]_0^{\Delta_i'} - \left[t + (b_j + l_i + t)e^{1 - \frac{b_i + t}{l_i}}\right]_{\Delta_i'}^{\Delta_i}
\end{aligned}$$

where $l_i = l_i(s)$. The integral in Equation (6) can be calculated analogously by

$$\alpha_j = \int_{\Delta_j'}^{\Delta_j} g\left(\frac{b_j - t}{l_j}\right)dt = \left[t - (b_j + l_j - t)e^{1 - \frac{b_j - t}{l_j}}\right]_{\Delta_j'}^{\Delta_j}.$$

3 AMM Details

3.1 The curvature parameter k

Each pool has its individual slippage function, defined through the curvature parameter k, see Section 2.4[6]. High values for k result in a low slippage for a wide coverage ratio range, which is suitable for pools with high volumes and without systemic drifts into one direction.

Low values for k result in a low slippage only in a narrow range around $\theta_i(s) = 1$, and significant slippage is applied when the pool leaves the equilibrium. This setup is best suited for pools with rather sporadic trades, and/or systemic drifts into one direction.

We will derive suitable initial values for k for all swap pools by evaluating historical volatility data for the different supported assets. Assets (i.e. fiat stables) that show a low price volatility (against the USD) can support high values, whilst currencies with a high price volatility, and/or a significant directional price drift should be set up with lower values.

[6] We neglect the parameter β in this section for clarity.

3.2 Rewards and penalties

As discussed in Sections 2.3 and 2.4, the following actions, from the perspective of a single pool, are rewarded or penalized:

	Deposit	Withdrawal	Swap-in	Swap-out
Coverage $\leq 100\%$	Reward $\uparrow$	Penalty $\downarrow$	Reward $\uparrow$	Penalty $\downarrow$
Coverage $> 100\%$	Neutral $\downarrow$	Neutral $\uparrow$	Penalty $\uparrow$	Neutral $\downarrow$

$\uparrow\downarrow$ indicate an increase/decrease of coverage ratio

Table 1: Overview of rewards/penalties

Generally speaking, deposits and withdrawals, as well as swaps into and out of a pool, have mirroring rewards and penalties. This way we achieve a self-sufficient incentive system.

Note that every new pool initially has a coverage ratio of 1 and in that moment all possible actions that drive the coverage ratio away from 1 will either lead to a penalty or be incentive-neutral. Bringing the coverage ratio back towards one either leads to a reward or is incentive-neutral.

3.3 Backstop pool

The backstop pool plays a central role in the design of Amber, as it takes on the swap pools' major risks. While the swap pool liability tokens represent the right to withdraw a certain previously deposited amount of X_i, the backstop pool LP tokens represents a share of the *total backstop liquidity*, which is defined as the sum of b_0 from the backstop pool, all surplus assets of X_i in pools with $\theta_i(s) > 1$ minus all shortfalls in pools with $\theta_i(s) < 1$. The total R-converted value is

$$\lambda := \max\left\{p_0 b_0 + \sum_{i\in\{1,\dots,n\}} p_i\gamma_i(s)\ ,\ 0\right\}.$$

The vector z of withdrawable balances is defined by the backstop pool balance and all pools with coverage ratio > 1:

$$z = [b_0, z_1, \dots, z_n]^T, \text{ where } z_i = \max\{\gamma_i(s), 0\}.$$

If a backstop pool liquidity provider j wishes to withdraw a share $\omega \in \left[0, \frac{e_0^T L e_j}{l_0(s)}\right]$ then the paid out amounts are given by

$$\underline{z} := \frac{\omega\lambda}{\|\pi(z)\|_1} z$$

where $\|\pi(z)\|_1 = \sum_{i=0}^{n} p_i z_i$ is the total R-converted value of z. Note that $\lambda \leq \|\pi(z)\|_1$ and that the total value of the paid out assets is $\|\pi(\underline{z})\|_1 = \omega\lambda$.

The backstop pool accepts deposits only in the reserve asset X_0, but the set of assets to be withdrawn depends on which swap pools have a coverage ratio above one. That is in addition to the backstop pool asset.

The LP can instead choose a different combination of assets, but for any increase of the X_0 part, the respective slippage applies. We expect most LPs to withdraw their funds either in full in X_0 despite the incurring slippage penalty,

or in a different single X_i (slippage penalty applies in case $\theta_i(s)$ should fall below 1). All users that choose a combination of assets that includes any assets other than X_0 have automatically passively contributed to the rebalancing of the backstop pool towards a higher coverage ratio. Users that withdraw their funds entirely in currencies other than X_0 contribute the most to this intrinsic rebalancing.

3.3.1 Backstop rebalancing

In order to further support the continuous rebalancing of the backstop pool, Amber will additionally support unidirectional trades between the backstop pool and any swap pool with $\theta_i(s) > 1$, but restricted to trades that increase $\theta_0(s)$. No slippage applies for these trades[7], and the trading fees are set to zero.

3.4 Protocol-owned Liquidity (POL)

So-called "mercenary capital" poses a major challenge for DeFi protocols: capital that always keeps watching out for the highest yield and quickly moves on to the next protocol once the typical initial period of high incentives comes to an end. As Amber rather targets low-risk capital for its swap pools, this issue should be less severe in our case. However, in order to make Amber increasingly independent from external LPs, the protocol is designed to own a significant and growing share of the liquidity in all its pools. This POL serves as a liability reserve that will stay in the pools forever (unless the DAO decides otherwise). The POL thus turns the protocol from a pure service provider for 3rd-party LPs into a vertically integrated end-to-end product.

3.5 Trading fees and LP incentives

Amber charges a certain percentage for each swap as trading fee f_i, which is deducted during a trade in both involved assets X_i. Initially f_i will be set to a fixed f for all assets, in later iterations f_i may be set to individual values for each pool, depending on their usage and risk parameters.

In contrast to most other AMM concepts, the trading fees on Amber are not allocated to the LPs, but are used to grow the protocol-owned liquidity (POL). LPs are instead compensated completely with $AMBR token emissions.

Trading fees are handled as a pool deposit on behalf of the POL in the respective pools where they occur. As the protocol has full ownership over the POL, this results in a backing of the protocol tokens with the accumulated trading fees. This "risk-free value" (RFV) should act as a valuation floor for the protocol token, and thus counters a potential sell pressure by the LPs. The POL will of course as well receive it's share of the protocol token emissions, and will use those tokens to vote for an evenly distributed minimum of token emissions for all active pools.

[7] otherwise we would open an exploitable security loophole.

4 Security considerations

4.1 Oracle security

The reliability of the oracles providing the pool asset's prices is vital to the security of Amber. The oracles must provide correct prices and update them quickly and reliably in case of price changes. The secure operation of the AMM thus depends on the quality and trustworthiness of the asset price oracles.

An inaccurate price provided by an oracle will allow trading assets at unfair conditions. This security risk can be mitigated to some extent by sanity checks on the data provided by the oracles, by using oracles with a secure multi-signature setup only and ideally by using more than one oracle for a single asset.

Frequent reliable price updates are also important in order to not trade using stale prices. The risk is somewhat limited for the Forex use case though, as most fiat currencies tend to be less volatile than the average crypto asset, thus causing less damage in case of stale prices.

It should also be noted that pools cannot easily be drained by trades at stale prices as the slippage will gradually increase as a pool's liquidity is decreasing. Any small discrepancy in asset pricing would only lead to a small exploitable amount, due to the imposed trading fee.

4.2 High-value contracts

Amber's pools, like any AMM pools, must be considered attack surfaces as they have access to third-party funds and are by their very nature exposed.

The backstop pool is likely the highest-risk asset as it is a central piece of Amber's architecture and holds more funds than the average swap pool. A successful attack on the backstop pool bears the risk of draining other pools as the backstop pool must be able to interact with all the swap pools.

A successful attack on any swap pool could affect any other pool's liquidity, too, as any swap pool can trade against any other swap pool.

In order to further strengthen the AMM's security, sanity checks can be added to compare pool balances after a trade or backstop withdrawal with the balances before. In case of a noticeable deviation the swap can be rejected. A simple version of these checks should be feasible at minor gas costs.

4.3 Swap pool bank run

Any swap pool's coverage ratio might deviate significantly from the ideal $\theta_i(s) = 1$. The worst case is a sudden significant decrease in liquidity, either by withdrawals when the pool coverage ratio is already low or by swaps of massive volume.

The main risks are the inability to swap to that asset if it is too illiquid or that the swap becomes too expensive due to the resulting high slippage. The other risk is that the price oracles might not update quickly enough, leading to arbitrage opportunities where there should not be any.

The risk of mercenary capital suddenly being withdrawn in significant amounts is reduced due to POL. It can be further mitigated by implementing and in-

centivizing an option for liquidity providers to lock their deposits for a certain amount of time or imposing a cool-off period prior to withdrawal.

4.4 Backstop pool bank run

Similar to the swap pools, the backstop pool might face a sudden outflow of capital as well. The caused effects are similar, but not the same.

As the backstop pool needs to cover various risks of the swap pools, a sudden drop in backstop liquidity can seriously impede the operational readiness of the AMM. POL and timelocks can mitigate that risk.

It is also possible that the backstop pool runs out of the backstop asset X_0, but still has access to significant swap pool surpluses. In this situation backstop LPs would be able to withdraw, but the backstop pool would be de facto inoperable as the swap pools expect the backstop pool to provide X_0 liquidity.

Those swap pool surpluses can, however, serve as a kind of "backstop to the backstop", allowing withdrawals from low-coverage swap pools even if there is no more backstop liquidity either, by withdrawing in some other swap pool's assets.

4.5 Backstop pool arbitrage loop

The backstop withdrawal option introduces additional complexity to the penalty / reward structure, which comes with the potential for systemic risks.

In order to mitigate this and render infinite loop attacks impossible that could exploit this complexity, we set the rewards for backstop rebalancing trades to zero (see Section 3.3.1). In addition there will be a "cool-off period" after depositing into a swap pool which will only allow withdrawing via the backstop pool after that time has passed.

5 Governance

Amber will broadly follow the governance model pioneered by Curve[11]. The protocol will be managed by holders of a protocol native governance token called Amber Governance Token ($AMBR). $AMBR tokens will represent voting rights. Holders will be able to decide on the ongoing allocation of the weekly $AMBR token emissions between the different pools. Furthermore, holders with enough $AMBR tokens will be able to make a formal proposal for change on the protocol. Token holders will then be able to vote on the proposal themselves or delegate their vote shares to a third party.

6 Outlook

In this section we present some feature ideas that we envision for future versions of the Amber protocol.

6.1 Self-learning curves

Once there is enough real-life trading data for Amber's swap pool, we plan to include an aspect of machine learning into a next iteration of the price curve design. Based on past trading patterns, we plan to derive the optimal curvature parameter k for each pool for keeping the pool in balance. For example, if a certain pool sees a clear pattern of constant outflows through directional trades, the sell-side of that slippage curve would go up quicker to proactively prevent the pool from getting depleted.

6.2 Permissioned pools

We plan to launch Amber permissioned pools in the future, which will be specifically designed for institutions in order to facilitate regulatory compliance in the DeFi space. Amber permissioned pools will only be accessible to whitelisted LPs and users, and thus allow, e.g., regulated money transfer services to utilize the power of DeFi for their cross-border Forex transactions for KYC'd clients.

6.3 Request for quote (RFQ)

A request for quote feature with a locked-in exchange rate for a certain amount of time would allow for an even more powerful integration of Amber into the money transfer flows. Whilst it's not feasible to realize an RFQ feature directly within the AMM logic, there's a great opportunity for a second-layer protocol on top of Amber to offer this feature - either in combination with a Forex hedging protocol, or solely for the permissioned pools, under certain terms and conditions, to prevent misuse.

References

[1] Hayden Adams. "Uniswap v1 Whitepaper". 2019.

[2] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, Dan Robinson. "Uniswap v3 Whitepaper". 2021.

[3] Fernando Martinelli, Nikolai Mushegian. "A non-custodial portfolio manager, liquidity provider, and price sensor". 2019.

[4] Michael Egorov. "StableSwap - efficient mechanism for Stablecoin liquidity". 2019.

[5] Ondo Finance Team. "Ondo Finance: Decentralized Investment Bank". 2021.

[6] Tokemak Team. "An Introduction to Tokemak". 2022.

[7] Mark B Richardson, Nate Hindman. "Bancor3 Primer". 2022.

[8] DODO Team. "A Next-Generation On-Chain Liquidity Provider Powered by Pro-active Market Maker Algorithm". 2020.

[9] Platypus Team. "Platypus AMM Technical Specification". 2022.

[10] A. Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017). DOI: 10.7717/peerj-cs.103.

[11] Curve Team. "Curve DAO Whitepaper". 2020.