Denaria Perpetual Exchange

v0.2

March 19, 2026

Contents

0 Introduction

Denaria has developed an innovative protocol for trading perpetual futures (perps). Perpetual futures are derivative contracts that enable users to speculate on the future price of an underlying asset without an expiration date. In our protocol, each perp pair consists of two assets: the primary asset—on which users base their price predictions—and a secondary asset used exclusively for accounting, serving as the quote asset and always represented by a stablecoin pegged to the dollar. Importantly, users never actually buy or sell the underlying asset; instead, their positions are represented by virtual tokens (vAsset for the underlying and vStable for the quote asset).

To begin trading, users first deposit collateral. In the initial version of the protocol, only stablecoins are accepted (with the list of approved stablecoins determined by the team at deployment). This collateral allows users to borrow virtual tokens with leverage. To take a position, a user exchanges the borrowed vTokens—for instance, borrowing vStable and converting it to vAsset to open a long position.

A key innovation of our protocol is the pricing mechanism for perps—the exchange rate between vAsset and vStable. Traditional approaches either use an oracle to fetch the underlying asset's price or employ an automated market maker (AMM) for virtual tokens (vAMM). We adopt a novel mixed approach: while our vAMM applies slippage to protect the liquidity pool from depletion, its pricing curve is continuously recalibrated around the actual market price sourced from an oracle. This hybrid model ensures that trades execute near the real market price, eliminating the need for additional incentive mechanisms (such as funding fees) to maintain price parity.

Risk management is integral to our design; if a user's collateral-to-position value ratio falls too low, liquidation is triggered.

Another standout feature of our vAMM is the flexibility it offers liquidity providers (LPs). Rather than requiring LPs to deposit liquidity in a fixed ratio dictated by the current state of the vAMM, our protocol allows deposits of vAsset and vStable in any proportion—even permitting one-sided liquidity positions. We have developed an efficient algorithm to accurately track each LP's share of both vAsset and vStable.

To foster market balance, we implement a funding payment mechanism that incentivizes users to take the less popular position. Unlike traditional models that calculate funding rates based on the difference between the mark and index prices, our approach determines the funding rate from the actual difference between the total values of short and long positions. This decouples trade

slippage from the funding rate, allowing us to offer both minimal slippage for traders and attractive funding payments for LPs, who typically hold the less popular positions.

1 Protocol overview

The protocol consists of several interconnected components: a multi-collateral Vault, the vAMM, a funding fees mechanism, trading and LP fee structures, and a liquidation mechanism. In this section, we provide a concise overview of each component and their interactions, with further details elaborated in the subsequent sections.

Users must deposit **collateral** to back their positions, and our protocol supports a multi-collateral system that allows deposits in various approved assets from a predetermined list. All collateral is securely held in a dedicated vault, and our approach offers enhanced flexibility compared to most multi-collateral protocols that impose rigid, hard-coded ratios among collateral types.

In our design, users have the freedom to choose their preferred proportions for depositing allowed collateral. This means that rather than being forced into a fixed collateral mix, traders can tailor their deposits to better match their risk management strategies. When a position is opened, the deposited collateral is locked in the vault. Profit and loss (PnL) payments are then distributed, under most cases, in the same ratios as the user's deposits, ensuring consistency between the collateral provided and the settlement of gains or losses.

However, our system also accounts for less typical scenarios. For example, if the vault predominantly holds collateral of type A and a new user deposits collateral of type B, the protocol intelligently adjusts the settlement process. In such a case, if the user realizes a profit, the payout cannot be made exclusively in type B. Instead, the system dynamically adjusts the collateral allocation and PnL payments based on a comparison between the vault's current ratios and the user's desired ratios. This dynamic adjustment mechanism allows the protocol to operate seamlessly under a wide range of conditions without the limitations imposed by fixed ratios.

In the first version, the allowed collateral assets will be limited to stablecoins pegged to the USD. These stablecoins are initially valued at face value (1 USD), simplifying the collateral valuation. Nevertheless, the protocol continuously monitors their market values, and if a deviation from the peg exceeds a predetermined threshold, special measures are implemented to safeguard the integrity of the vault. Detailed technical information regarding the design, risk management, and operational mechanics of the multi-collateral vault can be found in section 2.

Each perpetual type -that is, each derivative contract tied to a specific underlying asset- will have its own dedicated vault. Once a user deposits collateral into the vault, they can borrow vTokens, either vAsset or vStable, up to a maximum leverage set by the protocol. For the pricing of the underlying asset (and therefore of the vAsset we use an external oracle, but the price of vStable

is always assumed to be 1 USD) Importantly, borrowing vTokens alone does not open a position; users retain the option to repay these tokens at any time without committing to a trade. To open a position, a user must exchange some of the borrowed vTokens via the **vAMM**. For example, to initiate a long position on the underlying asset, the user would borrow vStable and then swap them for vAsset using the corresponding vAsset/vStable vAMM. When closing the position, the user reverses the process -exchanging vAsset back to vStable- with the difference between the initial amount of vStable borrowed and the vStable obtained upon closing representing the profit or loss (PnL). Funding payments, which are discussed later, should be also incorporated into the final PnL calculation.

Our vAMM is not a standard implementation; it leverages an external oracle to continuously capture the market price of the underlying asset and recalibrates its pricing curve (we use a AMM formula similar to that of the Curve protocol Curve [4],[5]) around this price. As a result, trades execute at the current market price plus a minimal, controlled amount of slippage, which is significantly lower than what is typically seen in conventional vAMMs. This design is more equitable for LPs, who consistently transact at the actual market price, and it importantly decouples the funding rate from the vAMM's slippage.

In traditional perpetual protocols that employ a static vAMM without external oracle feedback, the funding rate is derived from the difference between the mark price (the vAMM price) and an index price. This model incentivizes users to open positions on the less popular side to earn funding fees, thereby indirectly keeping the two prices aligned. However, for the funding rate to become significant under this approach, the gap between the mark and index prices must be relatively large, which corresponds to a high-slippage scenario on the vAMM. By decoupling the funding rate from slippage, our approach not only enhances predictability for users but also ensures that both metrics remain relatively independent.

Although conceptually distinct due to its reliance on an external oracle, our vAMM can be mathematically described as a shifted version of the Curve protocol's formula—centered around the actual market price. Nevertheless, we chose to redevelop the formulas from scratch rather than relying on the Curve code. Further details are provided in section 3.

The vAMM functions by enabling the exchange of vTokens only when liquidity is first provided by **LPs**. This introduces the challenge of accurately tracking the liquidity owed to each LP -that is, determining the amount of each token they can withdraw at any given moment, based on their initial deposits and all subsequent trades. This issue is particularly critical in a perpetual protocol, where an LP's liquidity directly influences their position size.

Traditional AMMs, such as Uniswap V2, require LPs to deposit both tokens in a ratio that matches the pool's current ratio. This requirement prevents deposits from altering the AMM price and simplifies liquidity tracking, but it is also quite restrictive. In our vAMM, we leverage an external oracle to continuously recalibrate the system, thereby avoiding the problem of imbalanced deposits affecting the price. Additionally, allowing LPs to provide imbalanced -

or even one-sided- liquidity gives them the flexibility to tailor their risk exposure. For instance, by depositing only vAsset, an LP effectively takes the risk of opening a passive short position, since traders will borrow (part of) his vAsset to open long positions.

This increased flexibility means that the balance of each vToken attributable to an LP changes with every trade. To manage this complexity, we have developed an algorithm that efficiently tracks each LP's balances using a single global variable combined with snapshots of this for each LP only whenever he interacts with the system. The details of this algorithm are provided in section (4).

Traders incur a **fee** for each trade executed within the vAMM—effectively applying a fee to opening positions, as opening a position involves a trade. These trading fees are collected in vStable. Additionally, while LPs have the flexibility to determine the composition of their deposited liquidity in the vAMM pool, appropriate fees are imposed to encourage deposits that help maintain balance in the pool. We have explored several fee mechanisms, which are summarized in section (5).

During an upward market trend -when an asset's price increases- it is natural for most users to open positions that follow the trend (long positions). Since LPs passively assume the opposite side of each trade, they can accumulate significant exposure and potentially incur losses if the trend persists. To mitigate this risk, our protocol implements a **funding payments** mechanism that compensates users who take the less popular position. By earning funding fees, these users are incentivized to open positions contrary to the prevailing trend, thereby reducing LP exposure.

Funding payments are calculated as the product of the position value and a funding rate. In conventional perpetual protocols that use vAMMs, the funding rate is typically proportional to the difference between the vAMM price (mark price) and an index price for the underlying asset. In those systems, funding payments help align the vAMM price with the real-world market price. In our protocol, however, price alignment is achieved through a different mechanism, and funding payments primarily serve to decrease LP exposure.

Because the standard funding rate formula is not effective for our approach, we have developed a new formula. We discuss the incentives and details of this new formula in section (6). There, we also explain how funding payments for each user can be computed efficiently using only a few global variables and snapshots taken at each interaction with the protocol -a challenging task, in particular for LPs, whose positions change continuously with every trade. Our solution leverages the LP balance-tracking matrix approach discussed earlier to accurately track these funding payments.

Position holders should support their positions with sufficient collateral. If a position's margin ratio—calculated as the ratio of the collateral value supporting the position to its total position value (including its PnL)—falls below a predetermined threshold, any party can trigger its **liquidation**. In this process, the liquidator effectively acquires the position at a discount, which acts as a fee for the liquidated party. Section (7) details the liquidation process in our protocol,

with special care taken to ensure that the discount is not negated by vAMM slippage, thereby maintaining a profitable incentive for initiating liquidations.

2 Multi stable-asset collateral vault

In this section we discuss the details of how the protocol can deal with multi-collateral. Although in the first version of the protocol only stable coins will be allowed as collateral, the fundamental principles we present here also apply to cases involving non-stable collateral, but additional considerations are required in such scenarios, which we will not address in this report.

The key issues to clarify fall into two categories:

- Economic Considerations:
  - Should all stablecoins be treated equally (e.g., should they all have the same maximum leverage)?
  - Should we impose restrictions on the composition of the total collateral pool?
  - How should the protocol handle a stablecoin that loses its peg?
- Accounting Considerations:
  - Should PnL accounting and payments be conducted in a single collateral type, or should they be distributed across all collateral types?
  - If the latter, how should the distribution be determined?
  - As a guiding principle, we will propose designs that allow users to withdraw their collateral in the same composition in which they deposited it when closing a position (We will see that this is almost impossible to be implemented under every possible circumstances). However, we cannot guarantee that profits will be distributed in the same manner.

Our system ultimately adopts the second approach described below. However, we have also evaluated an alternative method, and we will discuss each one in detail—outlining their respective advantages and disadvantages.

We denote the stable collateral assets permitted in our multi-collateral vault as Stable1, Stable2, ..., Stablen.

2.1 1st approach: Use a single collateral for paying the PnL

A quick search suggests that most perpetual protocols allowing multi-collateral follow a similar approach: they use a single collateral type (e.g., Stable1) for accounting and PnL payments, while other collateral types serve only to support positions and can be lost only in case of liquidation.

Detailed description:(we restrict ourselves to n=2 collaterals for simplicity)

- *Collateral Structure*
  - A user deposits y_1 amount of Stable1 and y_2 amount of Stable2, where either y_1 or y_2 can be zero.
  - The Stable1 deposit is added to the vault, while Stable2 is kept separately for each user.
- *Leverage and Borrowing Power*
  - The two collateral types can have different maximum leverage limits, $maxLev_1$ and $maxLev_2$.
  - Users can borrow up to $y_1 \cdot maxLev_1 + y_2 \cdot maxLev_2$ in USD-denominated virtual assets (vAssets).
  - Since Stable2 is used only to support positions and does not affect accounting, the relevant vAssets will be vStable1 and vAsset.
  - All accounting is conducted using Stable1.
- *PnL Payments and Withdrawals*
  - When a user closes a position, all PnL is paid in Stable1.
  - If the user's Stable1 balance minus PnL is negative, they are not allowed to withdraw any collateral until they deposit enough Stable1 to restore a positive balance.
  - One edge case remains: if a position is profitable but the vault lacks sufficient Stable1 to cover the payout. To mitigate this, we can impose restrictions on the overall collateral composition—for example, requiring that at least f% of the total deposited collateral be in Stable1. Deposits that would violate this threshold would be rejected.
- *Liquidation Mechanism*
  - In the event of liquidation, the system first liquidates the user's Stable1 collateral before using their Stable2.
  - The liquidation threshold for Stable2 could be set higher than that of Stable1, adding an extra safeguard.So the actual liquidation threshold for a user can be determined as

$$\begin{aligned} \textit{Liquidation threshold} = & \frac{y_1}{y_1 + y_2}(\textit{Liquidation threshold for Stable1}) + \\ & \frac{y_2}{y_1 + y_2}(\textit{Liquidation threshold for Stable2}) \end{aligned}$$

Advantages and Trade-offs:

- This approach ensures that all users pay and receive PnL exclusively in Stable1, with only liquidation profits being affected by the user's collateral composition.
- The main drawback is that even if a user deposits only Stable2, they must still hold some Stable1 to cover potential PnL, which is slightly inconvenient.

While this method may be ideal for protocols that prioritize a single collateral -such as Stable1 in our example- our protocol is designed to treat all stablecoins equally. To avoid the inconvenience of requiring users to hold a specific asset, we have decided against implementing this approach.

2.2 2nd approach: Flexible multi-collateral deposits and PnL payments

The user is permitted to deposit collateral in any of n stablecoins, denoted by Stable_1 to Stable_n . Let $y_1, ..., y_n$ the amount of each coin he deposits respectively as collateral. Some y_i can be 0. We define:

$$r_i = \frac{y_i}{y_1 + y_2 ... y_n}, i = 1, 2, ...n$$

All the stable coins are assumed to hold their peg. The sum in the denominator is the total value in USD deposited as collateral by the user.

If these ratios were the same for each user we could force the users to pay for their PnL in these same ratios and then the system would have the desired properties. But this would be too restrictive, so we will describe a more flexible approach.

We denote by $r_{i,tot}$ the same ratios for the whole system/vault before the new collateral deposit i.e if $y_{1,tot}, ...y_{n,tot}$ the total amount of Stable_1 to Stable_n tokens in the vault:

$$r_{i,tot} = \frac{y_{i,tot}}{y_{1,tot} + ... + y_{n,tot}} \tag{1}$$

and by $r_{i,tot}$ the total ratios after the new deposit i.e.:

$$r'_{i,tot} = \frac{y_{i,tot} + y_i}{(y_{1,tot} + y_1) + ... + (y_{n,tot} + y_n)} \tag{2}$$

- The user is free to deposit collateral in any ratio provided that:

$$\left| \frac{r_{i,tot} - r'_{i,tot}}{r_{i,tot}} \right| < th_i \tag{3}$$

  The thresholds th_i could be around 5-10%.

- When the user wants to close his position: Let's say that his total PnL are y USD (negative PnL means that the user should be paid and positive that he should pay). We compute $r'_i, i = 1, 2, ...n$ (check below) and give him back $y_1 - r'_1 \cdot y_1$ Stable_1 , ... $y_n - r'_n \cdot y_n$ Stable_n .

Computation of the ratios:If everyone was depositing collateral in the exact same ratios r_i, then r'_i should also be equal to r_i, meaning the user pays their PnL proportionally to their initial deposit. However, since not everyone deposits collateral at the same ratio, r'_i will depend on the user's initial ratio (r_i), as long as the total ratios of the pool immediately before ($r_{i,tot}$) their collateral withdrawal.

We first compute $dy_i = y \cdot r_i$ and then the total ratios after the withdrawal:

$$r'_{i,tot} = \frac{y_{i,tot} - dy_i}{(y_{1,tot} + dy_1) + ... + (y_{n,tot} + dy_n)} \tag{4}$$

Observe that the denominator equals $y' = y_{1,tot} + ... + y_{n,tot} - y$ and therefore does not depend on the splitting.

If

$$\left| \frac{r_{i,tot} - r'_{i,tot}}{r_{i,tot}} \right| < th'_i \tag{5}$$

holds for every i, we set $r'_i = r_i$ for every i. The withdrawal threshold th'_i could be the same or smaller than th_i.

If the above condition is not satisfied. We set $r'_i = r_{i,tot}$ for every i. If y is negative i.e. the position was profitable we are done. If y was positive, we have to do one extra check.

Starting from $i = 1$.

- 1. If $dy_i := y \cdot r'_i < y_i$ i.e. if the collateral deposited by the user initially at this stablecoin can cover the corresponding payment, continue with $i + 1$.
- 2. If not compute. Set $dy_i = y_i$, $y = -(dy_1 + ... + dy_{i-1})$ and apply the same algorith for the remaining coins with this new y.

Note: We are aware that the user can bypass the restrictions (3) and (4) by splitting his deposit or withdrawal to smaller ones that will gradually shift the total ratios of the vault. A mitigation against it is to use instead of the actual rations $r_{i,tot}$ time averages or even better to use not the ratios right before the action but the ratio before the last PnL settlement.

2.3 Handling stablecoin loss of peg

The discussion above assumes that all stablecoins used as collateral remain pegged to their face value. However, if a stablecoin loses its peg—an issue even when using a single stablecoin as collateral—additional measures are required.

One approach would be to use oracles to price stablecoins based on their actual market value rather than their face value. However, this would be excessive under most circumstances. Instead, we should just regularly monitor the prices of all stablecoins used as collateral. If they are normal we should use their face and not the market price. But, if the price of a stablecoin deviates significantly from its peg (e.g., by more than 0.05% for an extended period), we should:

- Prevent further deposits of that stablecoin as collateral.

- Require all payments to be made using that stablecoin to minimize the system's exposure.

3 Dynamic vAMM

In this section, we begin by describing the shifted constant sum and constant product formulas. Their simplicity helps illustrate how an AMM can be adjusted to track the market price, as well as the challenges that must be overcome for this approach to function properly. We then proceed to derive the formulas required for our vAMM.

In this section x and y are the amounts of the two assets in the pool (vAsset and vStable respectively for our vAMM case) and p is the market price (provided by an external oracle) of the first asset with respect to the second one.

The graphs for all the formulas discussed in this section can be found here: `https://www.desmos.com/calculator/ptnnvsv1eo`.

3.1 Constant sum AMM

The constant sum AMM is described by the following equation:

$$p(x - x_0) + (y - y_0) = 0 \tag{6}$$

where x_0, y_0 are the initial deposits.For every state (i.e. pair of x,y satisfying the equation of the AMM) the price is constant (therefore there is zero slippage on the swaps).The left had side of the equation is not dimensionless, but has units (liquidity in terms of the second asset). It would give us more flexibility to express the equation of the AMM in a form that involves a dimensionless invariant.

$$p(x - x_0) + (y - y_0) = 0 \Leftrightarrow \tag{7}$$

$$\frac{p(x - x_0) + (y - y_0)}{px_0 + y_0} = 0 \tag{8}$$

$$\tag{9}$$

We define the invariant of the constant sum AMM as:

$$S_{p,x_0,y_0}(x, y) := \frac{p(x - x_0) + y - y_0}{px_0 + y_0} \tag{10}$$

- $S_{p,x_0,y_0}(x, y) = 0$, iff (x, y) is a point of the AMM's graph
- $S_{p,x_0,y_0}(x, y) > 0$, iff (x, y) is a point above the graph
- $S_{p,x_0,y_0}(x, y) < 0$, iff (x, y) is a point below the graph

3.2 Constant product AMM

Starting with the constant product AMM (uniswap v2), we would like to keep the shape and general properties, but adjust it in a way that the graph passes through the point of the initial deposits (x_0, y_0) and the minus slope of the tangent of the graph at that point (i.e. the price as computed from the AMM) equals p. This can be achieved either by shifting and scaling the original graph along the x or the y axis. If we shift along the x axis we get the following formula:

$$[p(x - x_0) + y_0]y = y_0^2 \tag{11}$$

We divide by y_0^2 to make the formula dimensionless and we define the following invariant:

$$P^x_{p,x_0,y_0}(x, y) := \frac{[p(x - x_0) + y_0]y}{y_0^2} - 1 \tag{12}$$

- $P^x_{p,x_0,y_0}(x, y) = 0$, iff (x, y) is a point of the AMM's graph
- $P^x_{p,x_0,y_0}(x, y) > 0$, iff (x, y) is a point above the graph
- $P^x_{p,x_0,y_0}(x, y) < 0$, iff (x, y) is a point below the graph

There is an important distinction between the cases $px_0 < y_0$ and $px_0 > y_0$ (the case $px_0 = y_0$, corresonds to the standard constant formula of uniswap v2). In the first case, the graph intersects the y-axis. Therefore, it will work fine for short positions (that increase x), but not for longs (that increase y and therefore decrease x), because in that case the formula makes it possible to get a negative x as the solution. In the second case, the graph can be used both for longs and shorts, although for the case of long positions, since the graph is asymptotic to $x = x_0 - y_0/p$ i.e. the amount of the X asset cannnot decrease to an amount smaller that this value, therefore the AMM is not that efficient , since it does not allow to use all the x assets in the pool for swaps. The problem is particularly important when $x_0 >> y_0/p$.

Similarly, if we shift along the y axis

$$[y - y_0 + px_0]x = px_0^2 \tag{13}$$

We define another invariant:

$$P^y_{p,x_0,y_0}(x, y) := \frac{[y - y_0 + px_0]x}{px_0^2} - 1 \tag{14}$$

- $P^y_{p,x_0,y_0}(x, y) = 0$, iff (x, y) is a point of the AMM's graph
- $P^y_{p,x_0,y_0}(x, y) > 0$, iff (x, y) is a point above the graph
- $P^y_{p,x_0,y_0}(x, y) < 0$, iff (x, y) is a point below the graph

The distinction between the cases $px_0 < y_0$ and $px_0 > y_0$ is present also here. In the first case both shorts and longs are possible, but, since the graph is asymptotic to $y = y_0 - px_0$ this AMM is not that efficient in this case. In the second case, this AMM should be used only for long positions.

It should be clear that, although both of these shifted version of the constant product formula, have qualitative similar behavior are two different AMMs i.e. for the same input they give completely different outputs. For the reasons described above, we will use the first one for shorts and the second for longs.

In equations 11 and 13 if we divide by the left, instead of the right hand side, we get another invariant that describe the exact same graphs, but have some extra advantages that will be useful to us later.

$$P'^{x}_{p,x_0,y_0}(x,y) := 1 - \frac{y_0^2}{[p(x-x_0)+y_0]y} \tag{15}$$

$$P'^{y}_{p,x_0,y_0}(x,y) := 1 - \frac{px_0^2}{[y-y_0+px_0]x} \tag{16}$$

3.3 Dynamic Curve AMM

In this section, we derive from first principles the formula introduced by the Curve Finance protocol. Our version is a shifted adaptation of the standard Curve formula, centered around the actual market price.

We would prefer an AMM that combines the best properties of both the constant sum and constant product formulas. Specifically, we would like its graph to approximate the constant sum curve for values of (x, y) close to (x_0, y_0). This approach minimizes slippage for swaps that do not significantly move the pool, ensuring such trades are executed near the market price p. However, for larger swaps that shift the pool far from (x_0, y_0), significant slippage should apply, as with the constant product rule, to prevent the pool from being depleted.

This behavior can be achieved by defining a new invariant that is a linear combination of the constant sum and constant product invariants:

$$A \cdot S_{p,x_0,y_0}(x,y) + P^{x}_{p,x_0,y_0}(x,y) = 0 \tag{17}$$

or

$$A \cdot S_{p,x_0,y_0}(x,y) + P'^{x}_{p,x_0,y_0}(x,y) = 0 \tag{18}$$

for short positions and

$$A \cdot S_{p,x_0,y_0}(x,y) + P^{y}_{p,x_0,y_0}(x,y) = 0 \tag{19}$$

or

$$A \cdot S_{p,x_0,y_0}(x,y) + P'^{y}_{p,x_0,y_0}(x,y) = 0 \tag{20}$$

for long. A should be a positive either constant or function of the deposits i.e. of x and y and can also depend on the p, x_0, y_0 parameters. Observe that for $A = 0$ we get the (shifted) constant product formula and for $A \to \infty$ the

constant sum. The graph of this new AMM will lie between the graphs of the constant sum and that of the constant product.

We will use a function $A(x, y)$ inspired from Curve. Actually, our function is just a shifted version of the Curve function:

$$A(x,y) = \frac{Ay_0^4}{\left(By_0^2 + y_0^2 - [p(x-x_0)+y_0]y\right)^2} \tag{21}$$

then the equation for shorts becomes:

$$\frac{Ay_0^4}{\left(By_0^2 + y_0^2 - [p(x-x_0)+y_0]y\right)^2} \cdot \left(\frac{p(x-x_0)+y-y_0}{px_0+y_0}\right) + \left(1 - \frac{y_0^2}{[p(x-x_0)+y_0]y}\right) = 0 \tag{22}$$

which can also be written as a polynomial equation of degree 3:

$$ay^3 + by^2 + cy + d = 0 \tag{23}$$

where:

$$\begin{aligned}
a &= \lambda^3 \\
b &= \frac{Ay_0^4}{px_0+y_0}\lambda - 2\lambda^2 By_0^2 - 3y_0^2\lambda^2 \\
c &= \lambda\left(\frac{Ay_0^4}{px_0+y_0}(p(x-x_0)-y_0) + (B+1)^2 y_0^4 + 2(B+1)y_0^4\right) \\
d &= -y_0^6(B+1)^2 \\
\lambda &= p(x-x_0)+y_0
\end{aligned}$$

For longs we can use:

$$A(x,y) = \frac{Ap^2x_0^4}{\left(Bpx_0^2 + px_0^2 - [y-y_0+px_0]x\right)^2} \tag{24}$$

$$\frac{Ap^2x_0^4}{\left(Bpx_0^2 + px_0^2 - [y-y_0+px_0]x\right)^2} \cdot \left(\frac{p(x-x_0)+y-y_0}{px_0+y_0}\right) + \left(1 - \frac{px_0^2}{[y-y_0+px_0]x}\right) = 0 \tag{25}$$

which can be written as a polynomial equation:

$$ax^3 + bx^2 + cx + d = 0 \tag{26}$$

where:

$$
\begin{aligned}
a &= \lambda^3 \\
b &= \frac{Ap^2x_0^4}{px_0 + y_0}\lambda p - 2\lambda^2(B+1)px_0^2 - px_0^2\lambda^2 \\
c &= \lambda\left(\frac{Ap^2x_0^4}{px_0 + y_0}(y - y_0 - px_0) + (B+1)^2p^2x_0^4 + 2p^2x_0^4(B+1)\right) \\
d &= -p^3x_0^6(B+1)^2 \\
\lambda &= y - y_0 + px_0
\end{aligned}
$$

We will also need the "inverse" formulas of the above. Starting from (22) and solving for x we get the following polynomial equation:

$$ax^3 + bx^2 + cx + d = 0 \tag{27}$$

where:

$$
\begin{aligned}
A' &= \frac{Ay_0^4}{px_0 + y_0} \\
\lambda &= y - y_0 - px_0 \\
k &= y_0 - px_0 \\
a &= p^3y^3 \\
b &= A'yp^2 + 2p^2y^3k - 2p^2(B+1)y_0^2y^2 + (ky - y_0^2)p^2y^2 \\
c &= A'py(k+\lambda) + (B+1)^2py_0^4y + pk^2y^3 - 2(B+1)py_0^2y^2k + 2py^2k(ky - y_0^2) - 2(B+1)y_0^2yp(ky - y_0^2) \\
d &= A'yk\lambda + (ky - y_0^2)(B+1)^2y_0^4 + (ky - y_0^2)k^2y^2 - 2(B+1)y_0^2yk(ky - y_0^2)
\end{aligned}
$$

If the state of the pool before the trade was (x_0, y_0) and someone wants to get exactly dy vStable ($y = y_0 - dy$), he should add to the pool dx vAsset ($x = x_0 + dx$), and this dx can be found by solving equation (27). Also, we can use this equation to compute the amount of vAsset a user should add to the pool to get the amount of vStabe he needs to close his long position.

Starting from (25) and solving for y we get the following polynomial equation:

$$ay^3 + by^2 + cy + d = 0 \tag{28}$$

where:

$$
\begin{aligned}
A' &= \frac{Ap^2x_0^4}{px_0 + y_0} \\
\lambda &= p(x - x_0) - y_0 \\
k &= px_0 - y_0 \\
a &= x^3 \\
b &= A'x + 2x^3k - 2p(B+1)x_0^2x^2 + (kx - px_0^2)x^2 \\
c &= A'x(k+\lambda) + (B+1)^2p^2x_0^4x + k^2x^3 - 2(B+1)px_0^2x^2k + 2(kx - px_0^2)kx^2 - 2(B+1)(kx - px_0^2)px_0^2x \\
d &= A'xk\lambda + (kx - px_0^2)(B+1)^2p^2x_0^4 + (kx - px_0^2)k^2x^2 - 2(B+1)px_0^2xk(kx - px_0^2)
\end{aligned}
$$

If the state of the pool before the trade was (x_0, y_0) and someone wants to get exactly dx vAsset ($x = x_0 - dx$), he should add to the pool dy vStable ($y = y_0 + dy$), and this dy can be found by solving equation (28). Also, we can use this equation to compute the amount of vStable a user should add to the pool to get the amount of vAsset he needs to close his short position.

3.4 Using our dynamic vAMM to execute trades

Let's now discuss how these formulas are employed to execute trades. Recall that in the previous sections we used distinct formulas/graphs for short and long positions (they coincide only when $x_0/y_0 = p$). This was done to ensure that the resulting equations yield positive solutions and that the deposited liquidity is used optimally for trades.

For clarity -purely for exposition purposes without affecting the implementation- we combine these formulas and graphs into a single unified curve. This curve uses the formula for shorts when $x > x_0$ and the formula for longs when $x < x_0$, resulting in a graph that is both continuous and smooth, with a continuously differentiable first derivative. It can be described by an invariant $f(p, x_0, y_0, x, y)$, where the equation follows equation (22) or (25) for $x > x_0$, and equation (27) or (28) for $x < x_0$.

How should a trade be executed? Since we would like to execute trades as close to the market price as possible -a key motivation behind our vAMM design-, we could do the following. We start by retrieving the current pool deposits (x_0 and y_0) along with the current market price p from the oracle, and then solve the equation $f(x_0, y_0, p, x_0 + dx, y_0 + dy) = 0$, where dx, dy are the exchanged amounts.

In simpler terms, we continuously readjust the graph so that the initial state is always the current (x_0, y_0), ensuring that the price asymptotically aligns with the market price and slippage remains minimal. However, this dynamic mechanism introduces a potential exploit: a user could split a larger position into several smaller trades, thereby reducing overall slippage. In the extreme, by splitting a position into infinitely small trades, the user could achieve an execution price arbitrarily close to the market price (albeit still marginally higher).

To counter this exploit, we are imposing a minimum position size. Even without such a restriction, our dynamic vAMM ensures that LPs always receive at least the market price, thus protecting them from impermanent loss.

We are also implementing another measure to prevent this attack. Rather than constantly readjusting the graph with every trade around the current state, we adjust it so that two consecutive trades in the same direction (for instance, opening two long or two short positions) executed in quick succession have the same impact -actually slightly worse for the user- as a single trade whose size is the sum of the two trades. This approach renders the attack ineffective.

Let's discuss the specifics of how this works. For this method, the contracts will store three additional variables, denoted dx_0, dy_0, and t_0 (or $lastCurveUpdate$ in the contracts) and also the type of the last trade (long or short). Consider the case of long trades.A user executes a long trade (let's suppose that a long enough time has passed since the previous trade) by depositing dy vStable and getting back dx vAsset from the vAMM.We set t_0 to the current timestamp, $dx_0 := +dx, dy_0 := +dy$.We denote by x_0, y_0 the total amount of vAsset and vStable respectively in the vAMM after this trade. Let's assume the next trade is also long and its timestamp differs from t_0 by less than a specific threshold set in the contracts.

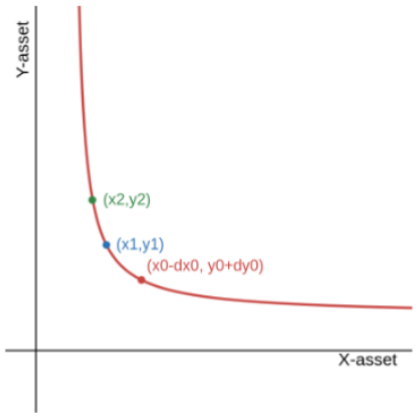

Figure 1: Pictorial representation of the mitigation against position splitting.

If the user exchanges dy' vStable, we first solve the equation $f(p, x_0 + dx_0, y_0 - dy_0, x_2, (y_0 - dy_0) + dy_0 + dy')$ for x_2 i.e. we align the graph around a state ($(x_0 + dx_0, y_0 - dy_0)$)as if the previous long trade has not happened and we just deposited the total vStable amount of all the last long trades $(dy_0 + dy')$ at once. The difference $dx_{tot} = (x_0 + dx_0) - x_2$ is the total amount of vAsset someone would get if he executed both trades at once (actually a little bit less, because of the slippage inrease after the first trade).The amount of vAsset the second trader gets is computed as $dx_{tot} - dx_0$. We finally increase dx_0, dy_0 by dx' and dy' respectively and set x_0, y_0 equal to the total vAsset and vStable liquidity in the pool after this second trade.

If the second trade was a short trade, or time more than the specified threshold had passed, we would reset dx_0 and dy_0 to 0 and would store this new timestamp as the new t_0.

4 Algorithm for efficiently tracking LP balances

We aim to allow imbalanced deposits, meaning LPs can contribute liquidity to the pool in proportions where the ratio of the two deposited assets does not necessarily match either the pool's current ratio or the market price. This flexibility allows LPs to implement more general strategies and select risk profiles that best suit their needs. However, this added flexibility introduces extra complexity into the system.

The LP's ownership of each asset is not constant. Consider an edge case where an LP deposits only one asset. In this scenario, the LP's holding of the other asset is initially zero. After a swap that utilises some portion of the deposited asset, the LP will also own a non-zero amount of the second asset. This change must be tracked accurately.

To make this approach feasible, we need an efficient method to compute these constantly changing balances in a way that is scalable and independent of the number of swaps or LPs. In these notes, we outline an algorithm to achieve this.

4.1 Description of the algorithm

Let's start by introducing some notation:

- t: discretized time, updated by 1 after each action (either swap or liquidity deposit/withdrawal).
- $x(t)$, $y(t)$: the total amount of vAsset and vStable in the pool, respectively, at time t.
- $x_i(t)$, $y_i(t)$: the amount of vAsset and vStable in the pool that belongs to LP i, at time t.

We will prove in the next section that each LP's balances evolve as a linear map of their balances at the time of deposit (we assume for convenience that the deposit happened at time $t = 0$):

$$\begin{bmatrix} y_i(t) \\ x_i(t) \end{bmatrix} = M(t) \begin{bmatrix} y_i(0) \\ x_i(0) \end{bmatrix} \tag{29}$$

The matrix $M(t)$ is global, the same for all LPs, so computing it efficiently lets us recover any LP's balances at any moment from a single stored snapshot.

Update rules for M. We will prove that M should be updated as follows, depending on the action.

Long swap (a user deposits dy vStable and receives dx vAsset from the pool):

Each LP's vAsset holding decreases in proportion to their current vAsset holdings, and each LP gains vStable in proportion to their vAsset holdings:

$$y_i(t+1) = y_i(t) + \frac{x_i(t)}{x(t)}\,dy \tag{30}$$

$$x_i(t+1) = x_i(t)\left(1 - \frac{dx}{x(t)}\right) \tag{31}$$

Defining $A_y(t+1) = \dfrac{dy}{x(t)}$ and $A_x(t+1) = \dfrac{dx}{x(t)}$, the per-trade matrix is:

$$A(t+1) = \begin{bmatrix} 1 & A_y(t+1) \\ 0 & 1 - A_x(t+1) \end{bmatrix} \tag{32}$$

Short swap (a user deposits dx vAsset and receives dy vStable from the pool):

Each LP's vStable holding decreases in proportion to their current vStable holdings, and each LP gains vAsset in proportion to their vStable holdings:

$$y_i(t+1) = y_i(t)\left(1 - \frac{dy}{y(t)}\right) \tag{33}$$

$$x_i(t+1) = x_i(t) + \frac{y_i(t)}{y(t)}\,dx \tag{34}$$

Defining $A_y(t+1) = \dfrac{dy}{y(t)}$ and $A_x(t+1) = \dfrac{dx}{y(t)}$, the per-trade matrix is:

$$A(t+1) = \begin{bmatrix} 1 - A_y(t+1) & 0 \\ A_x(t+1) & 1 \end{bmatrix} \tag{35}$$

In both cases the update rule for M is:

$$M(t+1) = A(t+1) \cdot M(t) \tag{36}$$

$$M(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{37}$$

For liquidity deposits and withdrawals, M is not updated (only the depositing or withdrawing LP's own snapshot changes, as described below).

Applying the update rule recursively:

$$M(t) = A(t) \cdot A(t-1) \cdots A(1) \tag{38}$$

Handling LPs that deposit after $t = 0$. If an LP deposits at time t_0, the relative matrix applicable to him at time $t \geq t_0$ is:

$$M_{t_0}(t) = A(t) \cdot A(t-1) \cdots A(t_0+1) = M(t) \cdot M^{-1}(t_0) \tag{39}$$

and his balances at time t are:

$$\begin{bmatrix} y_i(t) \\ x_i(t) \end{bmatrix} = M_{t_0}(t) \begin{bmatrix} y_i(t_0) \\ x_i(t_0) \end{bmatrix} = M(t) \cdot M^{-1}(t_0) \begin{bmatrix} y_i(t_0) \\ x_i(t_0) \end{bmatrix} \tag{40}$$

Steps for computing LP balances.

- At every swap we update the global matrix M using (38).
- When an LP deposits liquidity $(y_i(t_0), x_i(t_0))$ at time t_0, we store these deposited amounts and take a snapshot $M^{-1}(t_0)$.
- Whenever we need to compute the current balances of this LP, we compute $M(t) \cdot M^{-1}(t_0)$ and apply it to his stored initial balances via (40). No iteration over past trades or other LPs is needed.

Comment. This approach directly tracks balances rather than abstract shares. The overflow risk is naturally bounded: every LP's vStable and vAsset balances are bounded by the total vStable and vAsset in the pool, respectively. After many trades in the same direction, the entries of the A matrices are strictly less than 1 in the relevant row, so balances remain bounded.

The problem with many trades in the same direction is that the product $(1 - A_{x,1})(1 - A_{x,2}) \cdots \to 0$, driving asset balances toward zero. This is economically correct—LPs on the depleted side lose their holdings to the trader flow—and requires no special treatment in the accounting.

4.2 Proof that the algorithm correctly computes the balances

We prove (29) for every t by induction. For $t = 0$ this is trivial, since $M(0) = I$:

$$y_i(0) = 1 \cdot y_i(0) + 0 \cdot x_i(0) \tag{41}$$
$$x_i(0) = 0 \cdot y_i(0) + 1 \cdot x_i(0) \tag{42}$$

We assume the formula holds at time t and prove it holds at $t+1$. We study each action separately.

Long swaps. A user deposits dy vStable and receives dx vAsset. The pool's vAsset decreases by dx and its vStable increases by dy.

Each LP's vAsset holding decreases proportionally:

$$x_i(t+1) = x_i(t) - \frac{x_i(t)}{x(t)}\, dx = x_i(t)\left(1 - \frac{dx}{x(t)}\right) \tag{43}$$

Each LP earns new vStable proportional to its share of the vAsset pool (since it is the vAsset that is consumed by the trade):

$$y_i(t+1) = y_i(t) + \frac{x_i(t)}{x(t)}\, dy = y_i(t) + A_y(t+1)\, x_i(t) \tag{44}$$

Writing $[y_i(t), x_i(t)]^T = M(t)[y_i(0), x_i(0)]^T$ by the inductive hypothesis, and using $M(t+1) = A(t+1) \cdot M(t)$:

$$\begin{aligned} M(t+1)\begin{bmatrix} y_i(0) \\ x_i(0) \end{bmatrix} &= A(t+1) \cdot M(t)\begin{bmatrix} y_i(0) \\ x_i(0) \end{bmatrix} \\ &= \begin{bmatrix} 1 & A_y(t+1) \\ 0 & 1 - A_x(t+1) \end{bmatrix}\begin{bmatrix} y_i(t) \\ x_i(t) \end{bmatrix} \\ &= \begin{bmatrix} y_i(t) + A_y(t+1)\, x_i(t) \\ (1 - A_x(t+1))\, x_i(t) \end{bmatrix} = \begin{bmatrix} y_i(t+1) \\ x_i(t+1) \end{bmatrix} \end{aligned}$$

Short swaps. The idea is identical with x and y interchanged. A user deposits dx vAsset and receives dy vStable. Each LP's vStable decreases proportionally, and each LP earns new vAsset proportional to its share of the vStable pool:

$$\begin{aligned} M(t+1)\begin{bmatrix} y_i(0) \\ x_i(0) \end{bmatrix} &= \begin{bmatrix} 1 - A_y(t+1) & 0 \\ A_x(t+1) & 1 \end{bmatrix}\begin{bmatrix} y_i(t) \\ x_i(t) \end{bmatrix} \\ &= \begin{bmatrix} (1 - A_y(t+1))\, y_i(t) \\ x_i(t) + A_x(t+1)\, y_i(t) \end{bmatrix} = \begin{bmatrix} y_i(t+1) \\ x_i(t+1) \end{bmatrix} \end{aligned}$$

Liquidity deposits. When a new LP deposits dy vStable and dx vAsset at time $t+1$, the balances of all existing LPs do not change and M is not updated. The new LP's initial balance vector is simply the deposited amounts:

$$y_i(t_0) = dy \tag{45}$$

$$x_i(t_0) = dx \tag{46}$$

with $t_0 = t+1$, and a snapshot $M^{-1}(t_0)$ is stored. From (40), the balances of all existing LPs are unchanged, since their formula only involves $M(t) \cdot M^{-1}(t_0^{\text{old}})$ which is not affected by the absence of an M update.

5 Fees

5.1 Fees for opening/closing a position

In this section, we outline how swap/trading fees are applied, collected, and fairly distributed to LPs using the balance-tracking mechanism described in Section 4. Fees are always paid in the quote asset (vStable in our protocol) and are dynamically adjusted based on the imbalance between total long and short positions. This dynamic fee structure is designed to incentivize both LPs and traders to maintain system balance.

The Y-asset denotes the quote asset (vStable for our protocol), while the X-asset represents the base asset (vAsset in our case).

On each trade, a fee f should be applied. A portion ϕ of this fee will go to the protocol, while the remainder will be distributed to LPs in proportion to their balances. For example, if a user opens a \$100 position, they would pay $\$\,100 \cdot f$ in fees, of which $\$\,100 \cdot f \cdot \phi$ goes to the protocol, $\$\,100 \cdot f \cdot (1-\phi)$ is distributed to the LPs, and the actual value of the position opened would be $\$\,100 \cdot (1-f)$. Typical values for f range from 0.1% to 1%, and ϕ generally falls between 10% and 30%.

The fee f should be carefully chosen to make the protocol competitive relative to its competitors. Once this value is determined, all other parameters of the protocol can be optimized around the chosen fee structure.

To further enhance protocol stability, it would be beneficial for the fee percentage to be dynamic, helping the system remain (or move closer to being) delta-neutral. Each time a trader opens a position, the LPs are forced to take the opposing position. For instance, if traders consistently open long positions, the LPs will accumulate increasing short exposure.

We can track this exposure by maintaining the total difference between long and short positions opened by traders. This difference can be naively computed as the cumulative sum of dx, where dx is negative for shorts and positive for longs. A simple dynamic fee mechanism could involve defining two fees, f and $f' > f$. The higher fee will be applied when the total exposure (measured as the absolute value of the long-minus-short difference) exceeds a specified threshold. This threshold can be expressed as a percentage th of the total liquidity in the pool. This approach incentivizes traders and LPs to rebalance the system when it becomes imbalanced, helping to maintain a healthier, more neutral protocol.

Let's see how these fees should be collected and distributed. We treat the short and long cases separately. Initially the pool holds x vAsset and y vStable.

Long positions:

- The user gives dy vStable.
- The dy amount is split as $dy = dy_p + dy_{\text{LP}} + dy'$, where $dy_p = dy \cdot f \cdot \phi$ is removed from the pool and given to the protocol, $dy' = (1-f) \cdot dy$ is added to the pool and given to the trader as the effective swap input, and the remaining $dy_{\text{LP}} = dy \cdot f \cdot (1-\phi)$ are the LP fees that stay in the pool and are distributed to LPs.
- For the computation of the output vAsset amount we use only dy': we replace y by $y + dy'$ and x by $x - dx$ in the vAMM formula and solve for dx. This dx is the amount of vAsset given to the user.
- We update the pool balances to $x - dx$ and $y + dy_{\text{LP}} + dy'$.
- We update the global matrix M using the long-trade rule from Section 4, with:

$$A_y = \frac{dy_{\text{LP}} + dy'}{x(t)}, \qquad A_x = \frac{dx}{x(t)} \tag{47}$$

That is, A_y captures the net stable added to the pool per unit of vAsset (LP fees included), and A_x captures the vAsset removed per unit of vAsset. Each LP's stable balance grows in proportion to their vAsset holding, and each LP's vAsset balance shrinks proportionally.

- The vStable amount added to the pool ($dy_{\text{LP}} + dy'$) is greater than the amount used in the swap (dy') by the LP fee amount. We therefore rebalance the vAMM formula by replacing x_0, y_0 with their updated values and solving for the new curve anchor dy'' via the Newton method, using $dy_{\text{LP}} + dy'$ as the initial approximation.

Short positions:

- The user adds dx vAsset to the pool.
- We compute the gross output vStable dy by replacing x by $x + dx$ and y by $y - dy$ in the vAMM formula and solving for dy.
- The dy amount is split as $dy = dy_p + dy_{\text{LP}} + dy'$, where $dy_p = dy \cdot f \cdot \phi$ is removed from the pool and given to the protocol, $dy' = (1 - f) \cdot dy$ is removed from the pool and given to the trader, and the remaining $dy_{\text{LP}} = dy \cdot f \cdot (1 - \phi)$ are the LP fees that stay in the pool and are distributed to LPs.
- We update the pool balances to $x + dx$ and $y - dy_p - dy'$.
- We update the global matrix M using the short-trade rule from Section 4, with:
$$A_y = \frac{dy'}{y(t)}, \qquad A_x = \frac{dx}{y(t)} \tag{48}$$
That is, A_y captures only the net stable leaving the pool (the LP fee dy_{LP} remains, so the stable row does not shrink by the full gross amount). Each LP's stable balance shrinks proportionally to $dy'/y(t)$, and each LP's vAsset balance grows in proportion to their vStable holding.
- The vAsset amount removed from the pool ($dy_p + dy'$) is less than the amount used in the swap (dy) by the LP fee amount. We therefore rebalance the vAMM formula by replacing x_0, y_0 with their updated values and solving for the new curve anchor dy'' via the Newton method, using $dy_p + dy'$ as the initial approximation.

Note: The fees for a short position are such that f is the percentage of the total amount that will be kept as fees. If we want to compute the fees $dy \cdot f$ as a percentage of the amount given to the user $dy' = dy \cdot (1 - f)$, then this equals $\frac{f}{1-f}$. The same holds for longs.

Using the algorithm from Section 4, the vAsset and vStable in the pool that belong to LP i at any time t can be recovered as:

$$\begin{bmatrix} y_i(t) \\ x_i(t) \end{bmatrix} = M(t) \cdot M^{-1}(t_0) \cdot \begin{bmatrix} y_i(t_0) \\ x_i(t_0) \end{bmatrix} \tag{49}$$

where t_0 is the time of the LP's last deposit and $y_i(t_0), x_i(t_0)$ are the vStable and vAsset amounts the LP deposited at that time.

5.2 Fees for imbalanced liquidity deposits

The protocol we are designing allows for liquidity deposits in any ratio $r = Dy/Dx$, where Dx represents the amount of vAsset and Dy the amount of vStable deposited by an LP into the pool. This includes one-sided liquidity deposits ($r = 0$ or ∞). This flexibility is made possible by two key features of the protocol:

- **Dynamic vAMM**: This enables exchanges to occur near the market price, even when the ratio of total deposits in the pool differs from the market price.
- **Balance-tracking matrix mechanism**: This approach efficiently tracks the continuously changing balances of LPs for imbalanced deposits, as described in Section 4.

This flexibility is an advantage of our protocol, as it allows LPs to implement strategies fitting their risk profiles and market expectations. For example, by depositing one-sided liquidity (e.g., only vAsset), an LP primarily gains short exposure, since traders can only buy their vAsset (although, over time, as trades occur, the LP's balances will naturally shift, and they will gradually own a mix of both vAsset and vStable).

While this flexibility benefits LPs and allows for diverse trading strategies, it can also lead to imbalanced pools—where the ratio of total vStable to total vAsset in the pool deviates significantly from the market price. Although our protocol remains robust under such conditions, continuing to offer prices close to the market price for most trades, heavily imbalanced pools can result in higher slippage.

As we will explain, the market inherently assists in rebalancing the pool. Additionally, we propose a fee/penalty mechanism for LPs that incentivizes them to deposit liquidity in ratios that will move the pool closer to balance. Before discussing the specifics of the penalty mechanism, it is essential to understand how LP fees are collected and distributed.

5.2.1 LP fee distribution

Suppose that right before an LP deposit the pool holds x vAsset and y vStable, with market price p. An LP wants to deposit Dx vAsset and Dy vStable as liquidity. We denote by f_{LP} and ϕ_{LP} the fee coefficient the LP should pay and the fraction of that fee that goes to the protocol, respectively. The total USD value of the deposited liquidity is $p \cdot Dx + Dy$, so the total fee is:

$$\text{fee} = f_{\text{LP}} \cdot (p \cdot Dx + Dy) \tag{50}$$

The fee is always collected in vStable and deducted from the deposited Dy before the LP's liquidity enters the pool. If $Dy \geq$ fee, the LP deposits $Dy -$ fee vStable and Dx vAsset into the pool. If $Dy <$ fee, the LP deposits zero vStable into the pool and the shortfall fee $- Dy$ is recorded as additional vStable debt for that LP (to be settled on withdrawal).

In both cases the LP's debt is the full pre-fee amounts: Dx vAsset and Dy vStable (plus any shortfall). The LP's *initial balances* stored for the tracking algorithm are the net amounts actually placed into the pool:

$$y_i(t_0) = \max(Dy - \text{fee},\ 0) \tag{51}$$

$$x_i(t_0) = Dx \tag{52}$$

and a snapshot $M^{-1}(t_0)$ of the current global matrix is stored. Note that no normalisation by total pool size is needed: the balance-tracking algorithm operates directly on these amounts.

The portion of the fee that goes to the protocol, $\text{fee} \cdot \phi_{\text{LP}}$, is removed from the system. The remaining LP fee $f_{\text{fee}} = \text{fee} \cdot (1 - \phi_{\text{LP}})$ is retained in the pool as vStable and distributed to existing LPs proportionally to their current USD-valued balances. Concretely, the fee is split as:

$$f_y = f_{\text{fee}} \cdot \frac{y}{y + x \cdot p} \qquad \text{(allocated to stable holders)} \tag{53}$$

$$f_x = f_{\text{fee}} \cdot \frac{x \cdot p}{y + x \cdot p} \qquad \text{(allocated to asset holders, paid in stable)} \tag{54}$$

The global matrix M is updated to distribute these fees. The per-LP stable balance increases by f_y/y per unit of vStable held (stable holders' share) and by (f_x/x) per unit of vAsset held (asset holders' share, converted to stable at price p). Defining:

$$A_x = \frac{f_y}{y}, \qquad A_y = \frac{f_x}{x} \tag{55}$$

the fee distribution update is:

$$M(t+1) = A_{\text{fee}} \cdot M(t), \qquad A_{\text{fee}} = \begin{bmatrix} 1 + A_x & A_y \\ 0 & 1 \end{bmatrix} \tag{56}$$

That is, only the stable row (row 0) of M is updated. The total vStable in the pool increases by f_{fee}.

5.2.2 LP fees as incentives

Imbalanced deposits and market dynamics. Let's briefly examine the relationship between imbalanced liquidity deposits and market dynamics. Suppose we have an imbalanced pool, where $y/x \gg p$ (the ratio of assets in the pool is much greater than the market price). The primary cause of this imbalance could be either directional trading or imbalanced liquidity deposits.

If the imbalance is driven by trades, it likely means that most traders are buying vAsset from the pool, probably due to expectations that the price of the asset will increase. In this case, a new liquidity provider has an incentive to deposit more vAsset, reducing the y/x ratio. This is advantageous because, with most traders buying vAsset, the LP can earn more fees. However, if the LP shares the traders' expectation of a rising asset price and wants to avoid taking on a short position, depositing only vStable is not a viable solution. Traders are unlikely to buy these vStable and will continue purchasing vAsset, leaving the LP with little opportunity to earn fees.

LPs should primarily expect to profit from fees, and their profits will generally be higher in non-directional markets (although our dynamic vAMM design minimizes potential losses in directional markets).

On the other hand, if the imbalance results from large, imbalanced deposits rather than directional trades, and trading activity is balanced, a new LP has no reason to make a similarly imbalanced deposit. Balanced deposits allow the LP to profit from fees in both trading directions, maximizing their earning potential.

In conclusion, while we cannot claim that market dynamics will always drive LPs to balance the pool, there appear to be no economic incentives for further imbalancing it.

Imbalanced pools and funding rate. The primary advantage of our dynamic vAMM approach is its ability to execute trades near the actual market price—albeit slightly higher due to slippage. This design provides significant protection for LPs against impermanent loss. As a result, the funding rate, when calculated in the traditional manner, becomes negligible.

In the new funding rate mechanism we are designing, we aim to incorporate the discrepancy between y/x (the pool's ratio of assets) and p (the market price). By accounting for this difference, the funding payments will naturally help reduce pool imbalances caused by trading activity.

Imbalanced pool and slippage. If the ratio of assets in the pool (y/x) and the market price differ substantially, the result will be higher slippage for traders. We discuss the relation between imbalanced pools and slippage in the appendix. For most cases the increase in slippage is not as severe as one might expect. We suggest the simple fee mechanism described above and expect that this, combined with the market incentives (swap fees) and the funding rate, will help keep the pool balanced.

We compute the absolute difference

$$\left|\frac{y}{x} - p\right| \tag{57}$$

before ($\mathrm{dif}_{\mathrm{old}}$) and after ($\mathrm{dif}_{\mathrm{new}}$) the new LP deposit. If $\mathrm{dif}_{\mathrm{new}} < \mathrm{dif}_{\mathrm{old}}$ we apply a fee coefficient $f_{\min}$ (that could be 0 and should be less than the swap fee coefficient), otherwise we apply $f_{\max}$. We could use a more complex formula, e.g. a continuous function of the ratio $\mathrm{dif}_{\mathrm{new}}/\mathrm{dif}_{\mathrm{old}}$, but the simple two-value formula will probably be sufficient.

6 Funding payments

An essential feature of any perpetual protocol is its funding payments mechanism. Different protocols employ distinct formulas for computing funding rates (a brief collection of the most popular methods can be found in [1]).

Despite the differences in implementation, all funding mechanisms share two core characteristics:

- *Direction of Payments*: If the funding rate is positive, long positions pay short positions; if negative, short positions pay long positions. This is just a convention.
- *Balancing the System*: Funding payments are designed to help balance the system.Although, what it means "balancing of the system" varies for each protocol.

Let's discuss the second characteristic in more detail. It is often stated that funding payments help keep the perpetual contract price (also called the *mark price* in perpetual protocol terminology or the mid-price in AMM language) close to the price of the underlying asset (referred to as the *index price*).For instance, if the perpetual price is higher than the price of the underlying, this suggests an excess of traders opening long positions. In such cases, the funding rate becomes positive, requiring long position holders to pay funding fees to short position holders, who represent the less popular side of the market. This mechanism incentivizes traders to take positions that reduce the price imbalance, helping align the perpetual price with the market price.

Maintaining this alignment is crucial for any perpetual protocol. Traders use perpetual contracts to bet on the future price of an asset relative to its current price. Their decisions are based on expectations about the market, so if the protocol's perpetual prices deviate significantly from real-world market prices, it contradicts traders' expectations and makes the protocol less appealing.

In our protocol, however, we employ a different mechanism to keep the perpetual price (mark price) close to the market price of the underlying asset. As a result, if the sole purpose of funding payments were to maintain this alignment, we wouldn't need to include them in our design.

That said, the fact that we can keep these two prices closely aligned even without relying on funding payments is a significant advantage. It allows us to maintain lower funding payments, giving us a competitive advantage over other protocols.

However, it's important to note that funding payments serve other, perhaps even more critical, roles beyond price alignment.

As previously mentioned, funding payments incentivize traders to take the less popular position in exchange for collecting funding fees. This dynamic particularly benefits LPs, who provide liquidity passively and inherently take the opposite position to the trader's position.In non-directional markets, funding payments (along with trading fees) are expected to yield positive profits for LPs,

moving them from a neutral state (neither profits nor losses) to a profitable one. In strongly directional markets, these payments also help reduce LP losses, reducing their exposure.

Another critical role of funding payments is reducing the system's overall exposure, which in practice affects the LPs. When traders open long positions, they buy vAssets from the pool, which are provided by LPs. Consequently, the LPs passively take the opposite side of the trade—in this case, selling vAssets and effectively assuming a short position.

The greater the imbalance between total long and short positions taken by traders, the higher the system's exposure, as LPs are forced to take a position equal in magnitude to the imbalance but with the opposite sign. Funding payments are designed to mitigate this imbalance. They achieve this by incentivizing traders to take the less popular position to earn funding fees and also by disincentivizing further imbalance, as traders are discouraged from taking the more popular position due to the increasing funding costs.

The "total exposure" of the system can be defined in several ways, with each definition providing only a qualitative equivalence. The precise definition is critical for determining the funding payments and ensuring they function as intended.

At a broader level, funding payments are the mechanism that makes perpetual protocols viable. Unlike traditional futures contracts, perpetual futures (perps) do not have an expiration date. Without funding payments, a trader holding a long position would have little incentive to close it unless the price increased or they were at risk of liquidation due to a significant price drop.

Funding payments act as a recurring fee that continuously adjusts the health of open positions. This mechanism forces traders to take proactive measures, ensuring that perpetual protocols remain functional and balanced over time.

6.1 Funding payment definitions

In its most general form the fundyng payment a position holder -either a trader or an LP- should make (or receive) is given by the formula:

$$\textit{Funding fees for the position} = \textit{position value} * \textit{Funding rate} \tag{58}$$

where

- positive funding fees means that the position should pay these fees and negative that the position should receive the fees.
- position value: equals position size * price, and is positive for long and negative for short positions
- Funding rate: it is a global quantity (the same for every position), changes with time and trades and liquidity changes, is positive if longs should pay shorts and negative if shorts should pay longs

Several issues should be clarified: how often these funding payments should be made, how the position size is computed and how we can be sure that the total funding payments are zero (therefore holders of one position type are just paying holders of the oposite type and the system is not needed to provide any extra funds),exact formula of the funding rate and how it relates to the total exposure of the system and helps its balance, how the funding fees can be effectively computed on-chain.

6.2 The time component of the funding rate

First, we need to determine how frequently the funding payments should be calculated and accounted for. To clarify, this does not necessarily mean that actual payments must occur immediately (although, it should be possible for a trader or LP with negative funding fees to pair with one or more position holders with a positive funding balance and claim their fees). The funding rate can be expressed as:

$$\textit{Funding rate} = \frac{\textit{Funding coefficient} \cdot \Delta t}{\textit{Funding Interval}} \tag{59}$$

where

- Funding coefficient: This is not a standard name. We have chosen this more general name and we specialize it depending on the exact definition we will finally choose e.g. in the standard approach this is just the Premium (i.e. Mark Price - Index Price). This should be a global quantity and update on each action (trade or liquidity deposit).
- Funding interval: When time equal to the Funding interval has passed, the total fees (that should be paid) equal (Funding coefficient * total value of the most popular position type). The standard choice for the funding interval is 1 day (computed in seconds), but we could decrease it to increase the rate.
- Δt: Funding payments could be either periodic (e.g. every 8 hours) or can occur continously e.g. after every action. The second approach is preferable because it protects the system from mainuplation, as opening/closing a position right before or after a funding payment. In this case Δt is the time, measured in seconds, since the last action. On each action first we compute the funding rate, using the old funding coeeficient and the time since the last interaction and then we update the funding coefficient.

Note:In the implementation, we encountered the challenge of scaling the funding rate using an appropriate multiplier to achieve the desired value. Instead of using arbitrary multipliers, this can be replaced with 1/Funding Interval, which has a clear and precise meaning.

6.3 Standard definition of the funding rate

The standard definition (this one is used by the Perpetual protocol, check [2] and [3] for a very nice exposition of the funding mechanism in the Perpetual protocol) for the funding rate is:

$$\text{Funding rate} = \frac{(\text{Mark price} - \text{Index price}) \cdot \Delta t}{\text{Index price} \cdot \text{Funding Interval}} = \frac{\text{Premium} \cdot \Delta t}{\text{Index price} \cdot \text{Funding Interval}} \tag{60}$$

and in this case the funding fees can be simplified as:

$$\text{Funding fees} = \text{Position size} \cdot \text{Premium} \cdot \frac{\Delta t}{\text{Funding Interval}} \tag{61}$$

These are the funding fees related to only one action. The total can be computed by keeping a global sum:

$$\sum_i \frac{\text{Premium}(t_i)}{\text{Funding Interval}} \cdot (t_i - t_{i-1}) \tag{62}$$

where

- t_i: time of the i-th action (trade or liquidity deposit/withdrawal)
- $\text{Premium}(t_i)$: the premium for the period (t_{i-1}, t_i)
- Each trader should have a snapshot G_0 of this sum at the time of the opening of his position. On each update of his position, we can easily compute the total payments for this position by multiplying the size (the size of a trader's position changes only if he interacts directly with the protocol to change his position) by $G - G_0$, where G is the current value of the sum. For LPs the computation is more involved, since their position size changes not only when they directly interact with the system, but also at every trade. But we can still compute effectively their liquidity at every moment, and therefore their position, using the balance-tracking matrix approach described in Section 4.

For the computation of the premium, and therefore of the funding rate, we need the mark price. For perp protocols that use a vAMM the mark price is the spot price at the current state of the liquidity pool, i.e. the (minus) slope of the graph at the point (x, y), where x and y are the amounts in the pool. For a dynamic vAMM this is always equal to the market (index) price, therefore the Premium and the funding rate are always zero. To overcome this problem, we should take into account also the slippage and use the actual exchange prices. The formula for the sum above should be altered as:

$$G(t_i) = G(t_{i-1}) + \frac{(Dy_i/Dx_i) - p(t_i)}{\text{Funding Interval}} \cdot (t_i - t_{i-1}) \tag{63}$$

$$= G(t_{i-1}) + \frac{\text{slippage}(t_i)}{\text{Funding Interval}} \cdot (t_i - t_{i-1}) \tag{64}$$

where

- t_i: time of the i-th exchange in the vAMM
- $p(t_i)$: the market price of the underlying asset at time t_i
- Dy_i, Dx_i: the exchanged amounts of vStable and vAsset respectively

Note 1: We could replace the slippage(t_i) term by a weighted (by the size of the exchanged amount) moving average.
Note 2: This approach will give an even higher funding rate if we use the approach described in Section 3 (Market price update mechanism).

6.4 Position sizes

Let's now address an issue that may seem trivial when discussed at a high level but can lead to ambiguities when delving into the details: the matter of how position sizes are accounted for and used in the computation of the funding rate.

If FF_j are the funding fees for a position j (this could be either a trader or an LP position), then the sum of the funding fees should be zero (total sum of positive fees should equal minus the total sum of negative, i.e. payments of funding fees should cancel out and there should be no need for extra funds). This should be true for every funding payment (after every action) and not only for total payments over a period. This is true iff:

$$\sum_j FF_j = \sum_j \text{Funding rate} \cdot \text{Value of position } j \tag{65}$$

$$= \text{Funding rate} \cdot \sum_j \text{Value of position } j \tag{66}$$

i.e. iff the sum of the position values (positive for long and negative for shorts) is zero.

The position value can be computed by multiplying the position size (that we have to define) by the price (average price over the time period between two consecutive actions—check the algorithms in the next section):

$$\sum_j FF_j = \text{Funding rate} \cdot \sum_j \text{Value of position } j \tag{67}$$

$$= \text{Funding rate} \cdot \text{price} \cdot \sum_j \text{Size of position } j \tag{68}$$

Therefore, the sum of the funding payments will be zero if and only if the sum of the position sizes is zero. This holds true because positions are opened by exchanging amounts in a vAMM. However, let's explore this further to clarify the details. A long trader will borrow vStable and exchange them in the vAMM for vAssets. Therefore at the end his vAsset debt will be zero (he didn't borrow any vAssets) and his vAsset balance will be the output of the vAMM. On the contrary a short trader will borrow vAssets and exchange them for vStable. In both cases the difference:

$$\text{Position size} = \text{vAsset balance} - \text{vAsset debt} \tag{69}$$

captures the size of the position, that will be positive for longs and negative for shorts.

	Debt	Balance
long trader vAssets	0	dx
long trader vStable	dy	0
short trader vAssets	dx	0
short trader vStable	0	dy

The debt of a position changes only if the owner of the position interacts with the protocol to close or update the position. This is true for both traders and LPs. The balance of a trader can also only change by a direct interaction of the trader with the protocol, therefore a trader's position has constant size (between two interactions of the user with the protocol) and this simplifies the computation of the funding payments as we will see in the next section. The balance, and therefore the position size, of an LP constantly changes, as traders open positions transforming the liquidity of the pool from vStable to vAssets and vice versa. But we can still effectively compute these balances using the matrix from Section 4:

$$x_j(t) = \left[M(t) \cdot M^{-1}(t_0) \cdot \begin{bmatrix} y_j(t_0) \\ x_j(t_0) \end{bmatrix} \right]_1 \tag{70}$$

where $[\ \cdot\ ]_1$ denotes the second component of the vector (the vAsset row), t_0 is the time of the LP's last deposit, and $y_j(t_0), x_j(t_0)$ are the vStable and vAsset amounts deposited at that time.

When an LP deposits liquidity in the pool, this can be split into two actions: borrowing vAssets and vStable from the protocol (these amounts are his debts) and depositing these amounts to the pool (these are his balances). Therefore immediately after the deposit (until the first trade) the position of the LP is zero (this holds under the assumption that there are no fees applied to LPs, therefore the LP deposits all the amounts he borrowed and therefore his balances initially equal his debts. Below we will discuss what happens when we also apply fees to LPs). Therefore initially all the positions, and therefore their sum, is zero. We will prove that each trade, although it adds new positions, keeps the sum equal to zero:

- **Long position**: The user borrows dy vStable, sends them to the pool and gets back dx vAssets. His position size is $+dx$. But this vAsset amount should be subtracted from the vAsset balances of the LPs. In total the LPs' balances were reduced by $-dx$, i.e. in a zero sum, we added and subtracted dx, therefore it remains zero. If the trader pays fees (in vStable), what that changes is the actual dx he will get for the same dy, but even in this case the total sum of the position sizes is still zero.
- **Short position**: The user borrows dx vAssets and sends them to the pool to get some vStable. His position size is $-dx$. Since he sent these dx vAsset to the pool, the vAsset balance of the LPs will increase in total by dx, i.e. two opposite positions were opened and the sum is still zero.

Although closing a position is in some extent mathematically equivalent to opening a position of the opposite type, some extra care is needed related to PnL settling (check also the Funding payments example notes).

- **Closing a long position**: The user holds a long $+dx$ (vAssets) position and his debt is dy vStable. He exchanges his $+dx$ vAssets (therefore his vAsset balance is decreased by dx but the LPs' balance is increased by dx, therefore the sum of the positions stays zero even after the trader closes his position) and gets dy' vStable. The convention we follow is that funding payments and PnL are positive if the user has to pay and negative if he should be paid. We compute the total losses (or profits if negative) of the trader:

  $$\text{Trader's total losses} = (dy - dy') + (\text{total funding fees}) - (\text{already settled PnL}) \tag{71}$$

  If this is negative (i.e. the position was profitable) we delete his debt, burn his vStable balance and he can withdraw all his collateral. Moreover he should get an extra amount of collateral equal to his total profits. This procedure called PnL settling can be done as described in the Funding payments example notes: we increase the settled PnL of the other user by the same amount (but will not affect the face value of his position) and also reduce his collateral. If it is positive then again we should settle his PnL by pairing with another user with negative. The only difference is that now collateral will be transferred to the other user's account.

- **Closing a short position**: The user holds a dx (vAsset) short position, i.e. he holds dy vStable and his debt is dx vAsset. To close his position he should exchange (part of) his vStable to get exactly dx vAsset. Let's say that dy' vStable should be exchanged to get the dx vAsset he needs. The total losses/profits of the trader are:

  $$\text{Trader's total losses} = (dy' - dy) + (\text{total funding fees}) - (\text{already settled PnL}) \tag{72}$$

  and the settling should be made as in the long case.

6.5 Position sizes and LP fees

The discussion above demonstrates that the total sum of positions always remains zero. When a trader opens or closes a position, the LPs passively take the opposite positions, ensuring that the sum of all positions remains zero, even when fees are applied to the trade. Additionally, we showed that when an LP adds liquidity, this does not result in opening new positions and, therefore, does not alter the sum of positions, which remains zero. However, this no longer holds if fees are applied to LP deposits, requiring extra caution.

An LP borrows x vAsset and y vStable and deposits them into the pool. By depositing the full amount, the LP's vAsset balance equals their vAsset debt, resulting in a net position of zero. But what if fees were applied?

Suppose a portion of the deposits, f_x and f_y respectively, is allocated as fees to existing LPs. The new LP still deposits x and y, but his actual balances become $(1 - f_x) \cdot x$ and $(1 - f_y) \cdot y$, since $f_x \cdot x$ and $f_y \cdot y$ were distributed to the old LPs. While the total sum of positions remains zero—because the vAsset debt increased by x, the old LPs' balances increased by $f_x \cdot x$, and the new LP's balance is $f_x \cdot x$ i.e. also the total balance increase is x equal to the vAsset debt increase—the new LP now holds a nonzero position equal to $-f_x \cdot x$, effectively a short position. Meanwhile, the old LPs acquire a long position.

This outcome is undesirable, as it means that fees immediately introduce risk by creating an open position. Moreover, the resulting position depends on the LP's choice of asset allocation. For instance, if the LP deposits only vStable (i.e., if x=0), no new positions are opened.

To avoid this issue we suggest LP fees to be paid in the from of vStable only. The debt of the new LP is x vAsset and y vStable and all these are deposited into the pool. The total USD value of this debt is $d = x \cdot p + y$. If f is the fee coefficient, then his vAsset balance is x and his vStable balance is $y - d$. If the LP hasn't borrowed any vStable he should be forced to borrow at least the minimum amount to cover this fee.

6.6 Using the total exposure for the computation of the funding rate

The discussion in the previous section suggests a new definition of the funding rate, one that can yield non-negligible values even in our case, where slippage is nearly zero. Moreover, this approach decouples the funding rate from slippage, allowing us to achieve both minimal slippage and a significant funding rate simultaneously—an ideal scenario. We suggest the following definition:

$$\text{Funding coefficient} = \frac{\text{Total exposure}}{c \cdot \text{Total liquidity}} \tag{73}$$

where

- c: a constant between 0 and 1 (percentage of the pool liquidity)
- *Total liquidity*: total liquidity in the pool. The exact definition is not really crucial. We just use it to be able to define when a certain exposure should be considered small or large. It can simply be computed as $p \cdot x + y$, where x and y are the amounts in the pool and p the market price.
- *Total exposure*: sum of long minus sum of short positions, for traders only. The discussion in the previous section shows why this is equal to minus the total position of the LPs. Setting the funding rate proportional to this, traders are incentivized to decrease the total exposure of the LPs.

Let's describe the algorithm for computing the funding payments of a trader and an LP in more detail. We denote by $F(t)$ the funding rate at time t (actually it will be the funding rate times the market price, i.e. funding fees = position

size × price × funding rate, and we store in F the product price times funding rate). The initial value of the funding rate should be zero, since there are no open positions by traders at the protocol initially. The funding rate should change after each trade and liquidity deposit/withdrawal; the LP's positions change after each swap or if the owner of the LP position deposits/withdraws; and a trader's position changes only when he updates it and not when other users interact with the protocol. After each trade we also compute the total exposure $\text{Exp}(t) = \text{Exp}_s(t) \cdot p$ (where Exp_s is the size in terms of vAssets of the total exposure, i.e. the balance of vAssets held by long traders minus the debt of vAssets of short traders) and total liquidity $L(t)$. We denote by T the funding interval.

Funding payments for traders:

- The user opens his position at t_0 and the previous action happened Δt seconds ago. The size of the position is Dx vAssets (positive for long and negative for short).
- We update the funding rate:

$$F(t_0) = F(t_0 - \Delta t) + \Delta F(t_0 - \Delta t) \tag{74}$$

$$= F(t_0 - \Delta t) + \bar{p}\,\frac{\text{Exp}_s(t_0 - \Delta t) \cdot \bar{p} \cdot \Delta t}{c \cdot L(t_0 - \Delta t) \cdot T} \tag{75}$$

  and take a snapshot $F(t_0)$, stored in the user's state. $\bar{p}$ is the average market price for the Δt time interval, obtainable from the price oracle. $L(t_0 - \Delta t) = \bar{p} \cdot x(t_0) + y(t_0)$.
- We update the total exposure size:

$$\text{Exp}_s(t_0) = \text{Exp}_s(t_0 - \Delta t) + Dx \tag{76}$$

- When, at some later time t, the trader decides to close or update his position, we first compute the current funding rate $F(t)$ and his total funding fees (positive means he has to pay, negative means he should receive funding payments) as:

$$\text{Total trader's funding fees} = Dy \cdot (F(t) - F(t_0)) \tag{77}$$

  These fees should be added to his PnL from price changes and then settled, by pairing the trader with another user with opposite-side unsettled PnL. We then make the changes in the pool and update the funding rate as described in the previous step.

Funding payments for LPs:

- The funding rate is computed and updated as in the case of traders'. When an LP deposits or withdraws liquidity, the total exposure does not change

(it is just redistributed among the remaining LPs), but the funding rate changes because of the change in total liquidity. The tricky part is that an LP's position changes not only when the LP actively interacts with the protocol, but also on every trade.

- If two consecutive actions happen at times t_{i-1} and t_i, then for this time interval the total position held by the LPs equals $-\text{Exp}(t_{i-1})$ and the funding payments corresponding to all LPs are:

$$FP_{\text{LP}}(t_i) = -\text{Exp}(t_{i-1}) \cdot \frac{\text{Exp}(t_{i-1}) \cdot (t_i - t_{i-1})}{c \cdot L(t_{i-1}) \cdot T} = -\frac{\text{Exp}^2(t_{i-1}) \cdot (t_i - t_{i-1})}{c \cdot L(t_{i-1}) \cdot T} \tag{78}$$

  Observe that this is always non-positive (i.e. LPs in total will always receive and not pay funding fees—although some individual LPs may happen to pay—) as expected.

- The extra difficulty in the computation of the funding payments of individual LPs, compared to traders, is that an LP's position changes not only when the LP actively interacts with the protocol, but also on every trade. We denote by $FP_j(t_i)$ the total funding payments of LP j up to time t_i (positive if he has to pay, negative if he should receive a payment).

  At each step the LP's vAsset balance in the pool is $x_j(t_{i-1})$ (recoverable from the matrix as described in Section 4) and his direct net vAsset position (from any separate trading activity) is e_j = vAsset balance (trader side) − vAsset debt (total). The incremental funding payment is:

$$FP_j(t_i) = FP_j(t_{i-1}) + \big(x_j(t_{i-1}) + e_j\big) \cdot \Delta F(t_{i-1}) \tag{79}$$

$$= FP_j(t_{i-1}) + \big[M(t_{i-1}) \cdot M^{-1}(t_0) \cdot v_j(t_0)\big]_1 \cdot \Delta F(t_{i-1}) + e_j \cdot \Delta F(t_{i-1}) \tag{80}$$

  where $[\,\cdot\,]_1$ denotes the second (vAsset) component of the vector and $v_j(t_0) = [y_j(t_0),\, x_j(t_0)]^T$ is the LP's deposited balance vector at snapshot time t_0.

- Summing over all steps and factoring, we can write the total cumulative payment as:

$$FP_j = G(t) \cdot M^{-1}(t_0) \cdot v_j(t_0) - G(t_0) \cdot M^{-1}(t_0) \cdot v_j(t_0) + e_j \cdot \big(F(t) - F(t_0)\big) \tag{81}$$

  where G is a global 1×2 row-vector accumulator updated after every action as:

$$G(t_i) = G(t_{i-1}) + \Delta F(t_{i-1}) \cdot M[1](t_{i-1}) \tag{82}$$

  with $M[1](t)$ denoting the second row (vAsset row) of M at time t, and $G(0) = [0,\, 0]$.

- This can be written more compactly as:

$$FP_j = \big(G(t) - G(t_0)\big) \cdot M^{-1}(t_0) \cdot v_j(t_0) + e_j \cdot \big(F(t) - F(t_0)\big) \tag{83}$$

- Therefore we should store a global row-vector accumulator G updated via (82). For each LP, when he deposits liquidity at time t_0, we take three snapshots: $G(t_0)$, $F(t_0)$, and $M^{-1}(t_0)$ (the latter is already stored for the balance computation from Section 4). Whenever we want to compute the total accumulated funding payments for the LP, we use formula (83).

  Implementation note: When evaluating (83) at a moment when a rate increment ΔF has been computed but not yet written to G, the effective $G(t)$ should include the pending increment, i.e. $G(t) \leftarrow G(t) + \Delta F \cdot M[1](t)$, before applying the formula. This ensures the fee accounts for the full time interval up to the current moment.

7 Liquidations

To maintain system solvency, positions must be liquidated when the ratio of collateral supporting a position to its value falls below a certain threshold. Liquidation refers to the process where a user's position is sold to another user at a discount. This discount, which can also be considered a liquidation fee, does not have to go entirely to the liquidator—it can be split between the liquidator and the insurance fund.

7.1 Liquidation criteria

A position becomes eligible for liquidation based on its margin ratio (MR), calculated as:

$$MR = \frac{account\ value}{position\ value}$$

where:

- *Account value*: The USD value of the deposited collateral minus the total PnL for the position and any funding payments.
- *Position Value*: The size of the position multiplied by the index price of the underlying asset. Here, the index price is a time-weighted average of the market price over a few minutes (around 10 minutes). We should avoid using the actual (spot) market price to avoid liquidations due to temporary price fluctuations.

We define several margin thresholds:

$$MM_1 > MM_2 > ... > MM_n$$

and corresponding percentages:

$$p_1 < p_2 < ... < p_n$$

If a position's margin ratio satisfies $MM_{i+1} \leq MR < MM_i$, then up to p_i percent of the position size can be liquidated.

For large positions, full liquidation by a single user may not be feasible. Therefore, partial liquidations are allowed, enabling multiple users to collectively liquidate portions of the position up to the total allowable size.

7.2 Liquidation steps

Liquidation of long and short positions are symmetric, but for simplicity we will describe them separately.

7.2.1 Liquidation of a long position

User A holds a long position of size dx_A with margin ratio $MM_{i+1} \leq MR < MM_i$ and therefore it is liquidatable up to size $dx_{max} = p_i \cdot dx_A$. User B decides to buy $dx \leq dx_{max}$ of it. Let's call d the liquidation discount. This can be either fixed or could depent on the margin ratio of A i.e.

$$d = \frac{d_i}{2}\left(1 + \frac{MM_i - MR}{MM_i - MM_{i+1}}\right)$$

where d_{i+1} should be higher than d_i and , if we prefer continuous increase in the liquidation fee as the MR increases, $d_{i+1} = 2 \cdot d_i$.

The steps for the liquidation are as follows:

1. We compute (using the formulas we have described in a previous report)user's A funding fees and add them to the total funding fees he should pay. We could also force him pay a fee to the insurance fund (equal to a percentage of the USD value of dx).

2. To incentivize liquidators, the purchase price of the position should be set so that, if they choose to close it immediately after acquisition, they secure a profit based on the discount d (excluding any trading fees associated with closing the position).

   We compute (using the vAMM formulas of a previous report) the amount of vStable dy' that we would get if we exchanged dx vAsset, without applying trading fees (i.e. the ampunt of vStable that would get someone holding a position of size dx). This is just for accounting and we do not add or withdraw anything from the vAMM pool. The amount user B should pay to user A to get his position is $dy = (1 - d) \cdot dy'$. (The idea is that if user B immediately closes this new position he will get -minus trading fees- dy'. Therefore he earned $dy' - dy = d \cdot dy'$).

3. If user A is an LP, his vAssets are deposited as liquidity in the pool, so we cannot immediately transfer them to user B. We should first force user A to withraw (part of) his vAsset liquidity from the pool (this will increase the price of vAsset, therefore the above estimation is an underestimation

for user B i.e. if he closes the newly acquired position immediately he will get get even more -excluding trading fees-).If dx is the whole position of user A, we force him to withdraw all his liquidity. If it is just a percentage of his total position, we force him to withdraw the same percentage of his vAsset liquidity i.e. an amount dx' s.t.

$$\frac{dx'}{\textit{total vAsset liquidity of user A}} = \frac{dx}{\textit{user's A total position}}$$

4. We implement the transfer/sale of the position ie. we change the vStable and vAsset balances of A and B respectively by $(dy, -dx)$ and $(-dy, dx)$.If user A was an LP, we redeposit his remaining $dx' - dx$ in the pool as liquidity taking new snapshots.

5. We take all the necessary snapshots for both users.

6. If the margin ratio of the liquidator is below a threshold we revert.

7. If user's A collateral cannot cover his losses (bad debt), the insurance fund should cover them.

7.2.2 Liquidation of a short

User A holds a short position of size dx_A with margin ratio $MM_{i+1} \leq MR < MM_i$ and therefore it is liquidatable up to size $dx_{max} = p_i \cdot dx_A$. User B decides to buy $dx \leq dx_{max}$ of it. Let's call d the liquidation discount. This can be either fixed or could depend on the MR, as before.

1. We compute user's A funding fees.

2. We compute the amount of vStable dy' that we should exchange in order to get dx vAsset, without applying trading fees . This is just for accounting and we do not add or withdraw anything from the vAMM pool. We apply the discount and get $dy = (1 + d) \cdot dy'$

3. If user A is an LP and not a trader, we first force him withdraw an amount dy'' s.t.

$$\frac{dy''}{\textit{total vStable liquidity of user A}} = \frac{dy'}{\textit{user's A total position}}$$

4. We change the vStable and vAsset balances of A and B respectively by $(-dy, dx)$ and $(dy, -dx)$ (if dy is larger than dy'', we just decrease dx).

   All the next steps are the same as in the case of a long position.

Note: If the margin ratio of a position drops below a very low threshold and noone is liquidating it, maybe we should consider the option to allow force closing this position to prevent it from further accumulating losses that could lead to bad debt.

References

[1] *Funding rates:under the hood*, https://medium.com/derivadex/funding-rates-under-the-hood-352e6be83ab

[2] *Block-based Funding Payment On Perp v2*, https://blog.perp.fi/block-based-funding-payment-on-perp-v2-35527094635e

[3] *How Block-based Funding Payments are Implemented On Perp v2*, https://blog.perp.fi/how-block-based-funding-payment-is-implemented-on-perp-v2-20cfd5057384

[4] M. Egorov, *StableSwap- efficient mechanism for stablecoin liquidity*, https://docs.curve.finance/pdf/whitepapers/whitepaper_stableswap.pdf

[5] M. Egorov, *Automatic market-making with dynamic peg*, https://docs.curve.finance/pdf/whitepapers/whitepaper_cryptoswap.pdf