# Asynchronous Autonomous Market Maker

Alexander Lindgren, Jorge Sanmiguel

June 2023

The Unit of Liquidity is a value abstraction which can facilitate direct asset-to-asset swaps both cross-chain and on-chain with the same liquidity. This is achieved without liquidity partitions increasing the amount of liquidity needed to serve large swaps or virtual liquidity increasing the market-making cost. The Unit of liquidity is not an intermediate token the user is exposed to or requires lock and mint bridges, it is the result of a computation based on customizable independent swap curves. It is shown that the Balancer invariant can be replicated in an asynchronous environment, while a flatter invariant can be fitted for stable-coin swaps.

## 1 Introduction

Today, autonomous market makers create liquid markets for all kinds of tokens. Current AMMs generally work by examining on-chain variables, like token balances, and swap amount to then produce a quote. This works when the AMM can access the required information atomically on a chain. However, it doesn't work in a cross-chain environment where information is only available asynchronously. This inherent limitation locks liquidity to specific chains and causes liquidity fragmentation.

Prior attempts have focused on how to adapt existing solutions or how to improve cross-chain state synchronisation. Examples include: aggregating liquidity into a synchronous environment [tho20], using an intermediate bridge token [HBB18], or improving state aggregation [ZPB22]. However, these solutions make tradeoffs: Moving liquidity off-chain further increases fragmentation and makes the liquidity unavailable for actors who demand on-chain atomic liquidity; intermediate tokens introduce undesired exposure to users and add uncertainty; and state aggregation dependents on liquidity partitions reduces the maximum trade size for each additional chain, it doesn't support atomic swaps, and is currently limited to stable-coins.

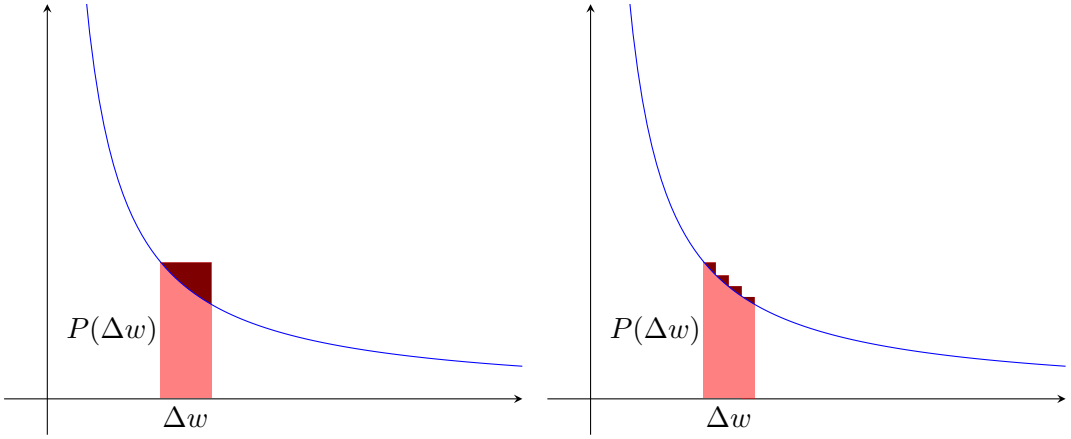
This paper introduces an autonomous market maker based on local invariants that are updated solely on local trade execution. Connecting the invariants defines a pool where assets can be swapped within, independently of where assets are located. Since each local instance is unaware of other instances, there is no state synchronisation nor an on-chain representation of the global state. This allows pools to scale linearly with the addition of more chains, in regards to both computation and liquidity. Furthermore, every previously mentioned issue has been solved, meaning: No undesired exposure, both stable-coin and volatile token support, all liquidity is available everywhere regardless of network size and all local liquidity can be used for atomic swaps.

## 2 Defining a Market Maker

The key idea behind automated market making is to price assets based on their current supply, $w$. To achieve this, define a decreasing[1], non-negative marginal pricing function, $P(w)$. The simplest way to evaluate the cost associated with a trade of size, $\Delta w$, is to multiply the current price by the trade size. Let $\epsilon(x)$ be some error function where $\lim_{x\to 0} \epsilon(x) = 0$, then the simple trade cost is given as:

$$U = \Delta w \cdot P(w_1) + \epsilon(\Delta w) \tag{1}$$

As $P(w)$ is a decreasing function, this approach only works for small $\Delta w$ as the strategy doesn't account for the price impact of changing the balance by $\Delta w$. The error's dependence on the trade size can be visualised by plotting the trade against a generic decreasing marginal price curve:




(a) No subdivsions (b) With subdivisions

Figure 1: The error of the simple approach

Using the dependence on trade size, subdividing the trade size into smaller amounts makes the error smaller.

$$U = \sum_{i=0}^{N} \frac{\Delta w}{N} \cdot P\left(w_1 + i\frac{\Delta w}{N}\right) + \epsilon\left(\frac{\Delta w}{N}\right) \tag{2}$$

The intuition can be seen in figure 1b. In the limit, as $N \to \infty$, the error incurred by the trade goes to 0 and the equation becomes the integral of $P(w)$ from $w_1$ to $w_1 + \Delta w$.

## 3 The Unit of Liquidity

The trade intermediate is the intermediary which market makers match assets against. In this paper, the trade intermediate will not be defined as an asset but rather as the trade cost, referred to as Units. Units are not globally shared but specific to a pool[2] of connected market makers. Since the trade cost is a number, Units can be abstracted

[1] While it is assumed the price function is decreasing, all results also apply to non-increasing price functions.

[2] A pool is defined as the largest set of market makers where at least one connection exists. If two pools contain two market makers that share a connection, the pools are sub-pools of a larger pool containing both.

away to allow the implementation to use existing message relayers to facilitate cross-chain swaps.
Given a pool of $n$ assets, $\{\alpha, \beta, \ldots, n\}$, define a decreasing, non-negative marginal price function for each asset: $\{P_\alpha(w), P_\beta(w), \ldots, P_n(w)\}$. The cost of trading an asset, i.e. the Units that the trade is worth, is measured as the integral of its price function. For some asset $i$, when exchanging $\Delta i$ for another asset, the Units are given by:

$$U = \int_{i_1}^{i_1+\Delta i} P_i(w)\ \mathrm{d}w \tag{3}$$

Within a pool, acquired Units, $U$, can be exchanged for any other asset. $U$ is computed from the reference of the initial asset, thus when evaluating the output the sign should be inverted. When 2 assets are matched within a synchronous environment, the swaps can be executed atomically. Examining a swap from $\alpha$ to $\beta$, the complete swap can found by solving for $-\Delta\beta$ as a function of $\Delta\alpha$:

$$\int_{\alpha_1}^{\alpha_1+\Delta\alpha} P_\alpha(w)\ \mathrm{d}w = -\int_{\beta_1}^{\beta_1+\Delta\beta} P_\beta(w) = \int_{\beta_1+\Delta\beta}^{\beta_1} P_\beta(w)\ \mathrm{d}w \tag{4}$$

### 3.1 Unit properties

Each integral represents a local invariant. Examine 2 series of swaps from a reference balance, $\alpha_0$, such that the total change in units is equal: $\sum_{i=1,\ldots,n} U_i = \sum_{j=1,\ldots,m} U'_j$

$$\begin{aligned}\int_{\alpha_0}^{\alpha_n} P_\alpha(w)\ \mathrm{d}w = \sum_{i=1,\ldots,n}\int_{\alpha_{i-1}}^{\alpha_i} P_\alpha(w)\ \mathrm{d}w = \sum_{i=1,\ldots,n} U_i = \\ \sum_{j=1,\ldots,m} U'_j = \sum_{j=1,\ldots,m}\int_{\alpha_{j-1}}^{\alpha_j} P_\alpha(w)\ \mathrm{d}w = \int_{\alpha_0}^{\alpha_m} P_\alpha(w)\ \mathrm{d}w\end{aligned} \tag{5}$$

Then it must hold that $\int_{\alpha_n}^{\alpha_m} P_\alpha(w)\ \mathrm{d}w = 0$. If $P_\alpha(\min\{\alpha_n, \alpha_m\}) \neq 0$ then $\alpha_n = \alpha_m$. Since $P_\alpha$ is decreasing, if it holds that $P_\alpha(\min\{\alpha_n, \alpha_m\}) = 0$ then $P_\alpha = 0$ in the interval $[\min\{\alpha_n, \alpha_m\}, \infty)$ and the difference has no value and can be ignored.
It has now been proven that using Units as an intermediate between chains allows swaps to be computed using only the local state, eliminating the need for state synchronisation. Furthermore, the above property guarantees that liquidity can be accessed asynchronously on each chain: While the execution order matters for traders, liquidity providers are indifferent[3].

## 4 Cross-chain Liquidity

While the integral construction has the asynchronous properties that are desired, it is not guaranteed to be a useful abstraction unless liquidity can be managed in a way which preserves the asynchronous properties.
To achieve this, the net debt, $U[i]$, distribution is tracked between assets. For each incoming and outgoing swap, the Units are added[4] to the counter. A reference value for each asset, $i_0$, can then be defined as the asset balance where at net-zero debt within the system:

$$U[i] = \int_{i_0}^{i_t} P(w)\,\mathrm{d}w \tag{6}$$

[3]Given that the fee is 0. If the fee is $\neq 0$ the order matters.
[4]Incoming swaps have a negative sign and thus are subtracted

With the introduction of net system debt and the reference balance, cross-chain liquidity can be defined via the reference balance. Let $\Delta i_t$ be the change in tokens and let $\Delta i_0$ be the change to the reference balance:

$$U[i] = \int_{i_0}^{i_t} P_i(w)\,\mathrm{d}w = \int_{i_0+\Delta i_0}^{i_t+\Delta i_t} P_i(w)\,\mathrm{d}w \tag{7}$$

This preserves the local invariant. Since $U[i]$ is constant outside swaps, when $i_t$ is increased from fees being deposited $i_0$ must increase. This makes it unsuitable as a direct record of ownership.

## 4.1 Vaults and Wrapping Vault Tokens

In practice, managing individual market makers is complicated. Instead, market makers are aggregated into vaults. Vaults are specific containers which hold tokens for market makers for each chain. Connecting vaults then define pools.
To track liquidity ownership, vault tokens representing shares of the reference value $i_0$ are used. The share of the reference value for each liquidity provider can be found by comparing their vault tokens, $pt$, with the total supply, $PT$.

$$\Delta i_0 = i_0 \cdot \frac{pt}{PT} \tag{8}$$

In other words, owning 20% of the vault $\left(\frac{pt}{PT} = 20\%\right)$ provides rights to 20% of $i_0$. To convert vault tokens into tokens, equation (7) should be used in conjunction with equation (8).

## 4.2 Simplifying Unit Accounting

Keeping track of $U[i]$ and $i_0$ on a per-asset basis is expensive for vaults with multiple assets. Instead, it is cheaper to keep track of the aggregated debt:

$$\sum_{i\in\{\alpha,\beta,\dots\}} U[i] = \sum_{i\in\{\alpha,\beta,\dots\}} \int_{i_0}^{i_t} P_i(w)\,\mathrm{d}w \tag{9}$$

The equation can be restricted by examining the single case of $\forall i : U[i] = 0$. Pick a reference asset, $j \in \{\alpha, \beta, \dots\}$, then the marginal price between $j$ and any other asset, $i$, is defined as: $\frac{P_i(i_t)}{P_j(j_t)}$. Using (5): $\forall i : U[i] = 0 \implies \forall i : i_t = i_0$ and the marginal price is known and constant as $P_{equal}$[5]. A relationship can be established between $j$ and any other asset:

$$P_{equal} = \frac{P_i(i_t)}{P_j(j_t)} = \frac{P_i(i_0)}{P_j(j_0)} \overset{\exists P_i^{-1}}{\implies} i_0 = P_i^{-1}\left(P_j(j_0) \cdot P_{equal}\right) \tag{10}$$

Assuming $P_i^{-1}$ exists[6], a closed-form solution to $i_0$ can be found. By combining (10) and (9), $i_0$ can be computed on-demand.

[5] $P_{equal} = \frac{P_i\left(i_{[U_i=0]}\right)}{P_j\left(j_{[U_j=0]}\right)}$ where the notation $i_{[U_i=0]}$ means the balance of the asset $i$ when $U_i = 0$

[6] $P_{equal}$ defines a point on the invariant. By always measuring local liquidity against this point, liquidity is distributed consistently across all prices.

### 4.3 Pool Invariants

Given a pool of $n$ assets, $\{\alpha, \beta, \ldots, n\}$, with balances $\{\alpha_t, \beta_t, \ldots, n_t\}$, the pool invariant is:

$$\begin{aligned} 0 &= \sum_{i \in \{\alpha,\beta,\ldots,n\}} \int_{i_0}^{i_t} P_i(w)\,\mathrm{d}w \\ \sum_{i \in \{\alpha,\beta,\ldots,n\}} \int^{i_0} P_i(w)\,\mathrm{d}w = K &= \sum_{i \in \{\alpha,\beta,\ldots,n\}} \int^{i_t} P_i(w)\,\mathrm{d}w \end{aligned} \tag{11}$$

The notation $\int^{i_t} P_i(w)\,\mathrm{d}w$ means the antiderivative of $P_i(w)$ evaluated at $i_t$. Notice that for the second equation, the left side is constant during swaps which is more similar to how CFMMs are usually written.
Examine an asset swap of asset $i$ to asset $j$, with $\Delta i$ given:

$$K = \int^{i_t} P_i(w)\,\mathrm{d}w + \int^{j_t} P_j(w)\,\mathrm{d}w \tag{12}$$

$$K = \left(\int^{i_t+\Delta i} P_i(w)\,\mathrm{d}(w) - U\right) + \int^{j_t} P_j(w)\,\mathrm{d}w \tag{13}$$

$$K = \int^{i_t+\Delta i} P_i(w)\,\mathrm{d}(w) + \left(\int^{j_t} P_j(w)\,\mathrm{d}w - U\right) \tag{14}$$

$$K = \int^{i_t+\Delta i} P_i(w)\,\mathrm{d}w + \int^{j_t+\Delta j} P_j(w)\,\mathrm{d}(w) \tag{15}$$

Where $U = \int_{i_t}^{i_t+\Delta i} P_i(w)\,\mathrm{d}(w) = \int^{i_t+\Delta i} P_i(w)\,\mathrm{d}w - \int^{i_t} P_i(w)\,\mathrm{d}w$. $\Delta j$ is defined based on U, as it would be for any swap from $i$ to $j$. The pool invariant is constant before (12), during (13) & (14) and after (15) a swap.

## 5 Price Curves

To materialize the theory behind Catalyst, suitable price curves that fulfil the design criteria outlined in Section 2 are to be found. Moreover, the price curve of choice must suit the underlying pricing characteristics of the vault's assets. Two pricing curves are proposed in this paper: a commonly used and well-known curve which allows the pool to serve any price for volatile assets, and one which serves a narrow range suitable for stable assets.

### 5.1 Volatile Assets

Asset pairs which aren't price bounded require a wide price range. For this, a non-linear price curve is selected:

$$P(w) = \frac{W}{w} \tag{16}$$

Where $w$ is the vault's asset balance and $W$ is a weight to adjust how liquidity is distributed within a connected pool of assets.

### Swap Equations

Let there be a Catalyst pool with tokens $\alpha$ and $\beta$, each weighted by $W_\alpha$ and $W_\beta$, and with vault balances $\alpha_t$ and $\beta_t$ respectively. The swap equations when swapping $\Delta\alpha$ tokens for $\Delta\beta$ tokens can be found by solving the generic swap equations introduced in Section 3:

$$U = W_\alpha \cdot \ln\left(\frac{\alpha_t + \Delta\alpha}{\alpha_t}\right) \qquad (17) \qquad\qquad \Delta\beta = \beta_t \cdot \left(1 - \exp\left(-\frac{U}{W_\beta}\right)\right) \qquad (18)$$

The equations can be combined to form a single full-swap expression:

$$\Delta\beta = \beta_t \cdot \left(1 - \left(\frac{\alpha_t + \Delta\alpha}{\alpha_t}\right)^{-\frac{W_\alpha}{W_\beta}}\right) \qquad (19)$$

It is recommended for full swaps to always be implemented via the split equations to align the mathematical accuracy across the different kinds of swaps.

### Swap Invariant

The invariant for the *Volatile* price curve can be found by using equation (11):

$$\begin{aligned} K_{vol} &= \sum_{i \in \{\alpha,\beta,\ldots,n\}} \ln(i_t) \cdot W_i \\ \exp(K_{vol}) &= \prod_{i \in \{\alpha,\beta,\ldots,n\}} i_t^{W_i} \end{aligned} \qquad (20)$$

The invariant can be rearranged such that the invariant of Balancer [MM19] is replicated. This implies that Catalyst is a further generalisation of the constant product market maker, enabling scaling of the familiar experience cross-chain.
Furthermore, since the invariant defines the balances of the market-maker, the market-making cost of the *Volatile* price curve is the same as the market-making costs of Balancer.
The way the asset weights influence the invariant curve can be seen in Figure 2.

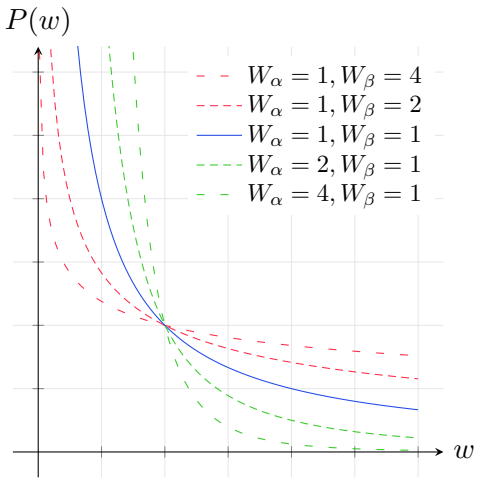



Figure 2: Volatile invariant curves for different weights combinations $(W_\alpha, W_\beta)$

### Liquidity Equations

Solving the vault debt requirement (7) for some asset $\alpha$ with the *Volatile* price curve yields the relationship described in (21), where $\Delta\alpha$ is the increase in the vault's asset balance $\alpha_t$, and $\Delta\alpha_0$ is the increase of the reference value $\alpha_0$.

$$\frac{\alpha_t}{\alpha_0} = \frac{\alpha_t + \Delta\alpha}{\alpha_0 + \Delta\alpha_0} \tag{21}$$

When depositing and withdrawing, the ratio between $\alpha_t$ and $\alpha_0$ must remain constant. The ratio can be rearranged to obtain an explicit relationship between the state of the asset $\alpha$ within the vault (i.e. $\alpha_t$ and $\alpha_0$), the change in the vault asset balance (i.e. $\Delta\alpha$, that is, the amount deposited/withdrawn), and the required change in the reference value $\Delta\alpha_0$:

$$\Delta\alpha_0 = \frac{\alpha_0}{\alpha_t} \cdot \Delta\alpha \tag{22}$$

Using (8), an expression that directly relates the vault asset balance change, $\Delta\alpha$, with the vault tokens change, $\Delta pt$, without the need for the net reference balance, $\alpha_0$, can be obtained:

$$\Delta pt = \frac{\Delta\alpha}{\alpha_t} \cdot PT \tag{23}$$

## 5.2 Stable Assets

The price curve of choice for stable assets is chosen such that its linearity can be adjusted via an *amplification* parameter $\theta$. This curve is given the *amplified* name, as it extends the curve introduced for *volatile* assets: in the limit, as $\theta \to 1$, the price curve tends to the *volatile* price curve (16). Hence, in the limit, the invariant of the *amplified* curve for $\theta = 1$ is the same as that of the *volatile* curve.
A weight $W$ can again be used to customize the price curve further. In this case, the weight has been set such that it can be used to adjust the ratio at which the stable assets trade.

$$P^{\theta}(w) = (1-\theta) \cdot \frac{W}{(W \cdot w)^{\theta}} \tag{24}$$

### Swap Equations

Following the same steps as for the *volatile* price curve outlined in Section 5.1, the swap equations for the *amplified* price curve can be derived:

$$U = (W_\alpha \cdot (\alpha_t + \Delta\alpha))^{1-\theta} - (W_\alpha \cdot \alpha_t)^{1-\theta} \tag{25}$$

$$\Delta\beta = \beta_t \cdot \left(1 - \left(\frac{(W_\beta \cdot \beta_t)^{1-\theta} - U}{(W_\beta \cdot \beta_t)^{1-\theta}}\right)^{\frac{1}{1-\theta}}\right) \tag{26}$$

Where $\Delta\alpha$ is the amount of token $\alpha$ sold for $\Delta\beta$ of token $\beta$. Again, the equations can be combined:

$$\Delta\beta = \beta_t \cdot \left(1 - \left(\frac{(W_\beta \cdot \beta_t)^{1-\theta} + (W_\alpha \cdot \alpha_t)^{1-\theta} - (W_\alpha \cdot (\alpha_t + \Delta\alpha))^{1-\theta}}{(W_\beta \cdot \beta_t)^{1-\theta}}\right)^{\frac{1}{1-\theta}}\right) \tag{27}$$

As with the *volatile* case, it is preferred to implement full swaps with the split equations.

### Swap Invariant

The invariant for the *amplified* price curve can be found by using equation (11):

$$K_{amp} = \sum_{i \in \{\alpha, \beta, \ldots, n\}} (W_i \cdot i_t)^{1-\theta} \tag{28}$$

To visualise the difference between the volatile invariant and the amplified invariant, the invariant can be plotted against the *volatile* invariant.

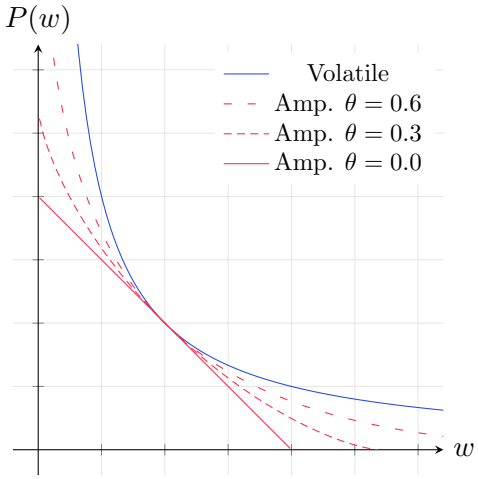



Figure 3: The amplified price curve invariant at different amplification values compared with the volatile price curve invariant.

Unlike the *volatile* price curve, the market-making costs for stable-coins is secondary since the value of the assets is roughly always equal in value while the absolute balance increases. Thus for market makers, liquidity usage is more important than the balance composition and the shallower invariant achieves exactly that: Much greater liquidity utilisation within a narrow price band.

### Liquidity Equations

The vault debt (7) resolves to (29) for the *amplified* price curve. From this equation, the relationship that deposits and withdrawals must obey between the vault's asset balance change, $\Delta\alpha$, and $\Delta\alpha_0$ (which ultimately translates to vault tokens) can be derived.

$$(W_\alpha \cdot \alpha_t)^{1-\theta} - (W_\alpha \cdot \alpha_0)^{1-\theta} = (W_\alpha \cdot (\alpha_t + \Delta\alpha))^{1-\theta} - (W_\alpha \cdot (\alpha_0 + \Delta\alpha_0))^{1-\theta} \tag{29}$$

## 6 Macro Equations

For large vaults, it might not be easy to acquire the exact token distribution that currently exists in the vault. As a result, users should be given the option to only deposit a subset of the tokens within the vault. This requires deriving an equation to convert units into vault tokens and vault tokens into units. There are different ways to find these

equations: Examining the local invariant from balance changes and swaps on the other side or solving the set of equations describing depositing and swaps.

$$\sum_{i\in\{\alpha,\beta,\ldots,n\}} \int^{i_t+\Delta i_t} P_i(w)\,\mathrm{d}w = \sum_{i\in\{\alpha,\beta,\ldots,n\}} \int^{i_t+\Delta i'_t} P_i(w)\,\mathrm{d}w \tag{30}$$

Where $\Delta i_t$ should be derived from (7) and $\Delta i'_t$ should be derived from (3) using $U_i = U \cdot w_i$ where $(w_i)_{i\in\{\alpha,\beta,\ldots,n\}}$ is a specific set of variables which maximise (30) for $\Delta\alpha_0$. For the chosen price curves, the macro swaps have been solved and can be read in the below table. Notice that the implementation details vary between the 2 derivations.

| **VolatileAssets** | **StableAssets** | |
|---|---|---|
| $pt = PT \cdot \frac{1 - e^{-\frac{U}{\sum_{i\in\{\alpha,\beta,\ldots\}} W_i}}}{e^{-\frac{U}{\sum_{i\in\{\alpha,\beta,\ldots\}} W_i}}}$ | $pt = \left(\left(\frac{N \cdot w\alpha_0^{1-\theta} + U}{N \cdot w\alpha_0^{1-\theta}}\right)^{\frac{1}{1-\theta}} - 1\right) \cdot PT$ | (31) |
| $U = \ln\left(\frac{PT}{PT - pt}\right) \cdot \sum_{i\in\{\alpha,\beta,\ldots\}} W_i$ | $U = N \cdot w\alpha_0^{1-\theta} \cdot \left(\left(\frac{PT + pt}{PT}\right)^{1-\theta} - 1\right)$ | (32) |

## References


[HBB18] Eyal Hertzog, Guy Benartzi, and Galia Benartzi. Bancor protocol. continuous liquidity for cryptographic tokens through their smart contracts, 2018. URL: `https://storage.googleapis.com/website-bancor/2018/04/01ba8253-bancor_protocol_whitepaper_en.pdf` [cited 26.04.2023].

[MM19] Fernando Martinelli and Nikolai Mushegian. A non-custodial portfolio manager, liquidity provider, and price sensor., 2019. URL: `https://balancer.fi/whitepaper.pdf` [cited 26.04.2023].

[tho20] Thorchain: A decentralised liquidity network, 2020. URL: `https://github.com/thorchain/Resources/blob/master/Whitepapers/THORChain-Whitepaper-May2020.pdf` [cited 26.04.2023].

[ZPB22] Ryan Zarick, Bryan Pellegrino, and Caleb Banister. Δ: Solving the bridging trilemma, 2022. URL: `https://www.dropbox.com/s/gf3606jedromp61/Delta-Solving.The.Bridging-Trilemma.pdf?dl=0` [cited 26.04.2023].