Understanding Balancer Pools

Angela Kreitenweis · Following
Published in Balancer Simulations · 12 min read · Mar 24, 2021

by *Angela Kreitenweis, Vasily Sumanov, Raul Martinez and Andrew Chiw*

Balancer is one of the leading DeFi protocols, recently surpassing $1B of total value locked. A growing number of DeFi projects use Balancer Pools as the core building block for liquidity mining, self-balancing portfolios, smart Defi indices, and other use cases.

> *Balancer's goal is to be the primary liquidity source in DeFi. We aim to accomplish this by providing the most flexible and simple protocol for automated asset management and decentralized exchange trading.*
> (Balancer)

Balancer Pools are built on a Constant Value Market Making algorithm, combining up to eight tokens for a wide array of liquidity provision strategies. Here, liquidity providers can deposit liquidity using only one token from the pool composition (single-asset liquidity provision), and the definition of weights provides additional flexibility for pool management.

This flexibility and composability come with a challenge for pool design. To leverage the full power of Balancer Pools, a project needs a detailed understanding of the core algorithm, the impact of design choices, and the resulting pool behavior. This is why the Token Engineering Community open-sourced a cadCAD model of Balancer Pools. This Python-based digital twin can be used for simulating and verifying the design of any Balancer Pool use case.

In this article, we will discuss the most important elements of Balancer Pools (V1) and their impact on pool behavior:

- core Balancer algorithms and transactions
- how transactions affect the system state
- why AMMs can be considered as *lazy* markets
- the role of arbitrage trading in Balancer Pools
- pool stability under normal and extreme market conditions
- Dynamic Weights Changing as a new approach for pool design

> *AMMs have quickly become the most successful building block in the DeFi space due to their simple and easy use. This stands in sharp contrast to how challenging yet fascinating they are to design.*

Core Balancer algorithms and transactions

The Balancer Pool AMM is defined by a function of the pool's balances and weights, constraining V to a *constant* ('Invariant V').

$$V = \prod_t B_t^{W_t}$$

Based on this value function, Balancer allows users to create pools with up to eight assets, user-defined weights, and customizable swap fees.

The value function uses normalized weights of the token, such that the sum of all weights is 1.

Balances and Price Definition

To understand how balances and prices are interconnected, we have to understand swaps first. Regardless of the pool's total number of assets, swaps in Balancer pools occur between only two token assets — a trading pair. A pool with eight token assets in the basket enables trades between 56 different trading pairs. The spot price of a swap is defined by balances and weights of the trading pair, with the constraint that no matter what exchanges are carried out, the share of the value of each token in the pool remains constant. At any pool stage, we can use this function to define the Spot Price.

SP_o_i is the Spot Price B_i is the balance of the token sold to the pool, W_i is the weight of the token sold to the pool, B_o and W_o are balance and weight of the token swapped out in return.

It is important to note that we get the price for an infinitely little trade size with this formula, since it ignores the change in balances due to the swap itself.
For a real trade and a given amount of *tokens in* (tokens sold into the pool), the price in *tokens out* is set so that the Invariant V remains constant after the trade. Hence, the change in balances has to be taken into account. Additionally, a fee might be charged:

A_o is the amount of tokens "out" a user will receive for a specified number of A_i (amount of tokens swapped in). B_i is the balance (before the trade) of the token swapped to the pool. W_i is the weight of the token swapped to the pool, B_o is the balance and W_o the weight of the token swapped out in return.

Weights and Price Definition

Weights define the *accounting* of an asset in a Balancer Pool, in other words the value distribution of the tokens in the pool. The following example illustrates how weights and prices are interconnected. We assume a token ratio of 1:10 in the pool, meaning by target Spot Price a token X should be 10x more valuable than token Y. By defining weights, the actual balance of a token X can be reduced, and fewer tokens Y are required to set up a pool with the same target Spot Price. This is particularly useful for liquidity bootstrapping.

B_x is the token balance of token X, B_y is the token balance of token Y, SP_x_y is the price of a token Y in token X.

Liquidity provision

One of Balancer Pools' most important features is the option to provide liquidity in only *one* token asset instead of the whole basket. This *single-asset liquidity provision* returns proportional pool shares. The number of pool shares is solely defined by the current supply of pool shares, the token balance and weight of the token *added*.

P_issued is the number of pool shares issued to a liquidity provider, P_supply is the current balance of pool shares (before liquidity provision), A_t is the amount of tokens added and B_t is the balance of tokens in the pool (before liquidity provision). Note that this formula applies only in case there is no swap fee.

This option to add liquidity improves the user experience for liquidity providers significantly. However, in contrast to a *full-basket liquidity provision* (liquidity provider supplies all assets in the pool), this transaction affects the ratio of token balances in the pool — and as a result the price! Like in a trade, a swap fee is charged.

How transactions affect the system state

The Balancer value function can be considered an output function engineered to encode the desired global properties of a Balancer pool, represented by the Invariant V.

The examples below illustrate how balances, weights, and prices are interconnected in a way that the Invariant holds throughout all system states. We start with an 80/20 token X/token Y pool at the same State A in all three scenarios.

Let's now observe how typical transactions impact token balances, weights, and the Spot Price in the pool in State B. For demonstration purposes, we ignore fees.

Scenario A: Full-Basket Liquidity Deposit

Observations:
1. In case of a full-basket liquidity provision, all token balances grow proportionally.
2. Since the liquidity deposit does not change the balance ratio in the pool, the Spot Price does not change.

Scenario B: Single-Asset Liquidity Deposit

Observations:
1. In a single-asset liquidity deposit transaction, the balances don't grow proportionally.
2. The balance ratio changes, and as a result, the Spot Price changes. Weights remain unchanged.

Scenario C: Swap

Observations:
1. Constraint by the Invariant V a token X is swapped for token Y. As a consequence, the balance of token X declines, while the balance of token Y grows.
2. The Spot Price changes due to the trade, weights remain unchanged.

Scenario D: Weights Change

Observations:
1. Weights in a Balancer Pool don't change due to a trade or liquidity deposit/withdrawal. However, as a parameter of the pool, they can be redefined.
2. A weight change always changes the Spot Price.

A variable Invariant

In contrast to what the term implies, the Invariant V changes in all scenarios, except for swaps, and only if there is no fee charged for swaps. Why this? First, we need to keep in mind that the value function represents the desired global properties of a Balancer pool. Let's consider the value function an 'energy conservation function', with energy being the tokens in the pool. It is absolutely critical that the Invariant V never goes up *unless* the energy it captures is added *from outside*.

There are two cases of adding energy. Adding tokens in liquidity deposits is *charging up* the pool for a right to discharge in the future (accounted in BPT). In case of a swap, fees are *charging up* the pool. Generally, a swap preserves the Invariant V, as shown in scenario C. However, in a regular pool, the swap fee charged is increasing the token price *on the way in,* which we can think of as a friction on the swap mechanism to ongoingly collect energy and make adding liquidity a viable long term investment — although there may be an impermanent loss.
There is a second element in the value function, the weights. A change in

weights is a major disruption to the pool's internal value distribution, which is reflected in the change of the Invariant V and prices. Thus, any weight change must be handled with care, and additional research is needed to avoid unintended side effects (see *Dynamic Weights Changing* below).

Using the value function as a tool for deriving and proving stability around an equilibrium is an approach widely used in control theory and dynamical systems theory. Here, we consider the value function as a state space representation of an economic system.

This chart shows the value function surface of a Balancer Pool after a series of swaps of equal trade size. In fact, the Invariant V changes with every swap, not due to the swap itself, though but due to fees increasing the pool's energy.

AMMs are Lazy Markets

In the previous section, we've outlined that Balancer Pools require swaps to change prices. Unlike at traditional exchanges, a token A's price in token B is not set within the traders' discretion. Instead, the price can only be changed by trades, which **requires liquidity**. This is why you could call AMMs *lazy markets*. Price changes need a token flow to adapt.
Let's explore this pool laziness with some examples.

Large pools are harder to balance

We already know that a price change requires a change in the token ratio of a particular trading pair. Large pools with huge liquidity require more capital to achieve an identical target token ratio, and thus, can be harder to balance.

The term 'price' in this example refers to the Spot Price.

Large pools are less sensitive to changes

In any Balancer Pool, the *size* of trade affects prices. With a given amount of tokens swapped into *different* pools, a trader ends with *different* amounts of tokens swapped out — depending on the pool liquidity.

The difference between the “ideal” Spot Price and the real swap price is called Slippage. In the example below, a trader swaps $10k of DAI for ETH in pools with different liquidity. With the same spot price in both pools, a trader swapping in Pool A would walk away with 5.7983 ETH and only 5.7362 ETH in Pool B — due to a significantly higher slippage rate.

This example is based on snapshots of Pool A and Pool B liquidity on Feb-12–2021

In this section, we’ve shown that pool size is an important property for pool behavior. Large pools require *more* capital for the same absolute price change, vice versa, they are less sensitive to swaps, and slippage is lower. This *laziness* can turn into a risk in extreme market conditions, as outlined below.

The role of external markets for Balancer Pools

With few exceptions, all assets traded in Balancer pools are also traded on external markets, such as centralized exchanges. Naturally, prices can be different in each market. This price spread between a Balancer Pool and an external market opens up a profit opportunity for arbitrage trades. Anybody can buy an asset at a Balancer Pool and sell it at a profit at an external market. These arbitrage trades are an important service to a Balancer Pool and a source for profit for liquidity providers, as stated in the Balancer whitepaper:

> *Balancer turns the concept of an index fund on its head: instead of paying fees to portfolio managers to rebalance your portfolio, you collect fees from traders, who rebalance your portfolio by following arbitrage opportunities.*
> (Balancer Whitepaper)

The example below shows a typical arbitrage trade between WETH, AAVE, and YFI, with a series of transactions across three exchange pools.

Source:
https://etherscan.io/tx/0xe4da12a319829711122c7741885c6d7404b73e02181b4873e2dfb199321bbc99

To calculate an arbitrage profit, slippage and swap fees must be taken into account:

A_o is the amount of tokens “out” a user will receive for a specified number of A_i (amount of tokens in). B_i is the balance (before the trade) of the token sold to the pool, W_i is the weight of the token sold to the pool, B_o is the balance, W_o the weight of the token swapped out in return.

A good example of two pools with equivalent token pairs constantly arbitraging against each other are the PERP pools on Uniswap and Balancer. As outlined above, pool liquidity matters. Large pools are more attractive for arbitrage trades since, given the same price spread, they offer greater absolute profits than small pools, and slippage is lower.

While arbitrage trading is important to rebalance a pool and converge to external price signals continuously, there are some negative effects, too. Arbitrage trades are draining value from the pool; tokens growing in value are constantly swapped *out* of the pool in exchange for tokens with decreasing value. As a consequence, rebalancing limits the value growth of a pool portfolio and the total value locked (TVL). We've discussed several alternative concepts for pool balancing in an earlier article and expect this topic to become a major Balancer Simulations research area in 2021.

Pool operation under normal and extreme market conditions

Under normal market conditions, Balancer Pools provide a robust environment for both trading and investment purposes. However, a significant price drop on external markets can pose a danger to the pool's capital. The following example paints a cascade of events caused by a token X dropping in price:

1) External (CEX) market price drops, and arbitrageurs take the opportunity to buy token X at a low price.

2) Then, arbitrageurs swap token X to a Balancer pool with a slower price reaction (lazy market) in exchange for a healthy token Y.

3) The pool in our example is large. It takes huge liquidity to close the price spread to external markets. Hence a lot of tokens X can be swapped *into* the pool.

4) More and more healthy tokens y are drained from the pool and the token ratio in the pool changes. Now, the pool enters the extreme ends of the constant product curve. Like in a vacuum, every remaining token Y swapped out of the pool sucks in even more cheap tokens X.

5) As a result, the pool ends up holding most of its liquidity in token X, and the liquidity providers' pool shares ($BPT) drop in value.

Of course, this situation can change again once the price for a token X will move back to normal. This is why the phenomenon is called *Impermanent Loss*. But what if the price will never go back to the "normal value"?

In the COVER protocol hack, the attacker minted an almost infinite number of COVER tokens. The price of COVER dropped promptly on CEXs, and the Balancer pool's capital was in danger. The PowerPool team had to freeze transactions and remove COVER tokens from the portfolio to ward the attacker off draining liquidity from the pool.

As these examples illustrate, AMMs can be difficult to handle under extreme market conditions. Pool managers need to continuously evaluate the total token supply on the market against the supply locked in a pool to ensure enough liquidity is available for rebalancing. Some projects may decide to implement emergency measures to protect a pool in extreme price movements, such as temporarily disabling swaps or implementing a *circuit break* based on tracking spot price deviation block-by-block.

Dynamic Weights Changing

Early versions of AMMs came with immutable token weights, maintaining a pre-defined Constant Value (the Invariant) throughout the pool's entire life cycle. As outlined above, some situations require specific adaptations to protect the pool's value. Dynamic Weights Changing provided some first promising results in applications led by Balancer and PowerPool. PowerPool developed and launched a Weights Changing Model processing native oracle price signals, focusing on gas efficiency and smoothness of weight transitions.

The idea behind *smooth weights changing* is to adapt weights gradually block-by-block, carefully driving arbitrage trades against external markets and pushing token balances to the new desired state.

In our example, a governance decision changes WBTC weight to 0.20 and BAL weight to 0.40 with a rate of 0.0001 per block. Over a period of roughly 3.69 hours, or 1000 blocks, weights are linearly adapted, while arbitrage trades continuously balance the pool against external markets.

Dynamic Weights Changing opens a range of promising opportunities for Balancer Pool use cases. Liquidity holders in pools with community governance can update portfolios in a secure and stable mode according to the ecosystem's objectives. Smart portfolios and indices can rebalance portfolios triggered by external price signals, M.Caps, TVL, or basically any other metric.

Dynamic Balancer pools can minimize value drain caused by arbitrage trades. In case of an external price change, weights can be adapted to minimize arbitrage opportunities and Impermanent Loss.

Over the last months, a variety of approaches on Dynamic Weights Changing has been introduced. We provide an overview in this article and invite all researchers in this field to join the **Balancer Simulations research group**, to contribute and benefit from open Balancer research.

Acknowledgments

Thanks very much to the following people for feedback: Markus Buhatem Koch, Michael Zargham, Ryan Watkins, Yan Liberman, and Andrew Penland.

About Balancer Simulations

Balancer Simulations is a project of TE-AMM and the Token Engineering Community, funded by grants from Balancer and PowerPool, and kicked off by EthicHub. It aims to build infrastructure and knowledge for rigorous Balancer Token Engineering to leverage the full power of Balancer Pools as a core building block in DeFi.

TE-AMM recently released the Balancer Pools V1.0 cadCAD model, a simulation framework for Balancer Pools. All research and models are available through open-source repositories and will be further developed by TE-AMM. We invite any project using Balancer Pools, or building products on top of it, to check out Balancer Simulations, join our Discord Channel, and the Balancer Simulations research group.

Token Economy Crypto Engineering Tokenomics Token Engineering

Written by Angela Kreitenweis

Following

516 Followers · Editor for Balancer Simulations

Founder of TE Academy, co-founded TokenEngineering Community. Establishing a new engineering discipline and crypto value flows for research and education.

More from Angela Kreitenweis and Balancer Simulations

Angela Kreitenweis in TE Academy

TE Fundamentals NFT Minting Seasons are here!

Mint your NFT certificate for free—Minting Season 1 starts on September 29

Vasily Sumanov in Balancer Simulations

Introducing Balancer Simulations

A Python-based simulation environment for Balancer Pools

4 min read · Sep 20, 2023

176

8 min read · Mar 9, 2021

340

Andrew Chiw in Balancer Simulations

On-Chain Data Ingestion for Balancer Simulations

How to turn on-chain events into simulation-relevant actions and inject real-world data...

7 min read · Apr 6, 2021

164

Angela Kreitenweis in TE Academy

Token Engineering Fundamentals

TE Academy builds the first comprehensive education & certification program in Token...

5 min read · Feb 4, 2022

573 3

See all from Angela Kreitenweis

See all from Balancer Simulations

Recommended from Medium

Mark Murdock in LayerZero Official

LayerZero V2 Deep Dive

Everything you need to know about V2: architecture, trust assumptions, tx lifecycle,...

32 min read · Jan 30, 2024

378 2

Modulus

Track 1: Upshot's ZKML Price Predictor

Debuting the largest AI application ever implemented on Ethereum — and yes, it's...

6 min read · Nov 21, 2023

113 4

Lists

Leadership

48 stories · 255 saves

Leadership upgrades

7 stories · 66 saves

Stories to Help You Grow as a Software Developer

19 stories · 865 saves

Good Product Thinking

11 stories · 484 saves

Juani in Balancer Protocol

Rate Provider manipulation in Balancer Boosted Pools scope...

Overview

8 min read · Sep 14, 2023

8

Artturi Jalli

I Built an App in 6 Hours that Makes $1,500/Mo

Copy my strategy!

· 3 min read · Jan 23, 2024

12.1K 148

Atis E

Protocol Fee Sharing and the Future of Uniswap

In the coming months, the Uniswap DAO appears poised to vote on protocol fee...

9 min read · 5 days ago

2

Ben Wee

Implications of a Uniswap Fee Switch

🦄 Overview

3 min read · Feb 25, 2024

10

See more recommendations