

# EulerSwap White Paper

Euler Labs

February 2025

## Abstract

EulerSwap is an Automated Market Maker (AMM) that uses unique features of Euler lending vaults to dramatically magnify capital efficiency using leverage. Unlike traditional AMMs that require fragmented liquidity pools, EulerSwap enables a single, cross-collateralized vault to provide liquidity across multiple asset pairs simultaneously, maximizing capital efficiency and scaling liquidity across DeFi. Using the Ethereum Vault Connector (EVC), EulerSwap allows market makers to borrow the out token of a swap using the in token as collateral, unlocking deep liquidity without requiring large static deposits. This liquidity-sharing model allows EulerSwap to achieve up to 40x the liquidity depth of traditional AMMs, efficiently repurposing idle assets for optimal capital utilization. At the core of EulerSwap is a highly customizable AMM curve that governs swap exchange rates, ensuring deep, just-in-time liquidity while dynamically incentivizing market balance. By combining on-demand borrowing, shared liquidity, and flexible AMM mechanics, EulerSwap eliminates inefficiencies in existing liquidity provisioning, offering superior depth, reduced costs, and greater control for liquidity providers.

## 1 Introduction

EulerSwap enhances trade execution by increasing the size of swaps that can be serviced compared to traditional AMMs. This is made possible by combining a novel AMM curve with the advanced [check deferrals](#) and [operator](#) features of the Ethereum Vault Connector (EVC). Market makers supply an initial amount of liquidity to a ‘swap account’ which can then be amplified through borrowing. Specifically, when a user swaps an ‘in token,’ EulerSwap borrows the corresponding ‘out token,’ using the ‘in token’ as collateral. Later, when a swap occurs in the reverse direction, the incoming ‘in token’ repays the outstanding loan, and any excess collateral is returned as the ‘out token,’ effectively closing the position. This leveraged approach significantly increases the available liquidity provided by EulerSwap and multiplies swap fees. It is somewhat analogous to MEV bots that provide just-in-time single-tick liquidity to Uniswap v3 in order to capture fees on large trades.

Beyond individual swap accounts, EulerSwap takes capital efficiency further by taking advantage of Euler’s modular design to allow multiple swap accounts to be built on top of the same lending vaults. This means that the same idle liquidity inside a single vault can provide depth for swaps across multiple asset pairs simultaneously. Much like Curve’s 3pool, but generalized to any number of asset pairings, a USDC vault could provide deep liquidity for 10+ correlated assets without requiring separate liquidity pools for each.

EulerSwap’s exchange rates are governed by a highly customisable AMM curve (illustrated in [Figure 2](#)) which incentivises swappers to maintain market maker positions in balance over time. The EulerSwap AMM curve is a novel construction blending constant-sum and constant-product models, allowing swap account creators to tailor their AMM parameters. Like [Stableswap](#), it supports concentrated liquidity for correlated assets, as well as [Uniswap v2](#)-style distributed liquidity for uncorrelated pairs. However, unlike those more traditional AMMs, EulerSwap allows asymmetric liquidity deposits, where different amounts of liquidity can be added to each side of the pool, and single-sided liquidity concentration, enabling deeper concentration of liquidity on one side of the pool relative to the other. This enables it to simulate the behaviour of atypical AMM protocols, like the MakerDAO [Peg Stability Module](#) (PSM).

For end-users, EulerSwap functions like a typical Uniswap-style interface. However, behind the scenes, it incorporates advanced mechanisms such as borrowing and repaying, custom pricing curves, and dynamic liquidity provisioning. The main target audience for liquidity providers in EulerSwap AMMs are profes-

sional market makers, token issuers, and other DAOs, whilst the main target audience for swappers are leverage traders on Euler, aggregators, intent solvers, and MEV bots.

## 2 Example

Suppose that Euler allows borrowing of USDT with USDC as collateral at a loan-to-value (LTV) ratio of 0.95, and vice versa. This means that for every \$1 of USDC or USDT collateral, a user can borrow up to \$0.95 of the other asset.

Now suppose you have a swap account with 1M USDC deposited as initial liquidity. Using maximum leverage your account could support deposits of 20M USDC and debts of 19M USDT. Alternatively, if you swapped your 1 million USDC to 1 million USDT, it could support the opposite. Let's say another user wants to swap 10M USDC for USDT. The steps are as follows:

1. They send 10M USDC to your swap account as the swap input.
2. Your swap account deposits the 10M USDC as collateral in Euler.
3. Your swap account borrows approximately 10M USDT against the account's collateral, which includes your original deposit, alongside the swap input.
4. It sends the borrowed USDT to the user as the swap output.

Importantly, this isn't a 1:1 swap because: a) you charge a fee for facilitating the swap; and b) the exact swap output is determined by an AMM curve, which factors in increasingly large price impact as the swap input amount increases. After the swap, your swap account now holds 11M USDC deposits and 10M USDT debt. It earns interest on the supplied USDC but must also pay interest on the borrowed USDT.

Later, when a swap occurs in the reverse direction, the incoming 'in token' repays the outstanding loan, and any excess collateral is returned as the 'out token.' The AMM curve dynamically adjusts incentives such that imbalances encourage swaps in the opposite direction, facilitating natural rebalancing. This mechanism ensures that positions do not remain open for extended periods, reducing exposure to borrowing costs. Over time, you will incur costs due to small interest rate differentials, whilst generating significant swap fees through utilisation of idle liquidity in the lending protocol. The whole process is depicted in Fig. 1.

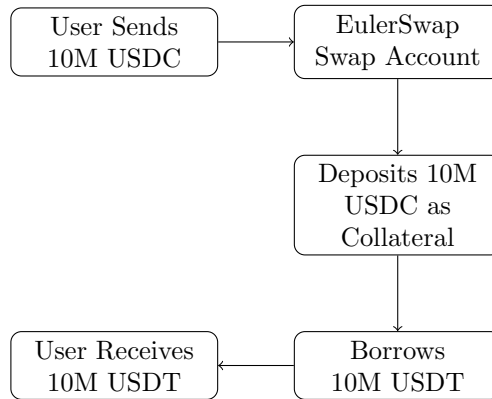


Figure 1: **Swap flow in EulerSwap AMM.** EulerSwap's just-in-time liquidity borrowing. The swap account dynamically increases liquidity by borrowing the "out token" against the "in token."

## 3 Curve

The space of possible reserves in a EulerSwap AMM is determined by how much debt a swap account is allowed to hold. The EulerSwap curve passes through an equilibrium point  $(x_0, y_0)$ , at which the marginal price is defined by:

$$\left. \frac{dy}{dx} \right|_{(x_0, y_0)} = -\frac{p_x}{p_y}. \quad (1)$$

Unlike most AMM curves, which are usually defined by a single convex function, EulerSwap uses a piecewise-defined curve, with different functions providing trading behaviour either side of the equilibrium point:

$$f(x) = \begin{cases} f_1(x), & 0 < x \leq x_0 \\ f_2(x), & x_0 < x \end{cases}. \quad (2)$$

In the domain  $0 < x \leq x_0$ , the curve is defined by

$$f_1(x) = y_0 + \frac{p_x}{p_y} (x_0 - x) \left( c_x + (1 - c_x) \left( \frac{x_0}{x} \right) \right). \quad (3)$$

In the domain  $x_0 < x$ , the curve is defined by

$$f_2(x) = \frac{\sqrt{\left( \frac{p_x}{p_y} (x - x_0) + y_0(1 - 2c_y) \right)^2 + 4c_y(1 - c_y)y_0^2} - \left( \frac{p_x}{p_y} (x - x_0) + y_0(1 - 2c_y) \right)}{2c_y}. \quad (4)$$

The curve coefficients in these equations,  $c_x, c_y \in [0, 1]$ , represent liquidity concentration parameters, determining how steeply the curve slopes to the left and right of the equilibrium point  $(x_0, y_0)$ . When  $c_x, c_y$  are close to 1, the curve behaves more like a constant-sum AMM, while for values close to 0, it behaves more like a constant-product AMM. A full derivation is given in the Appendix 5.1.

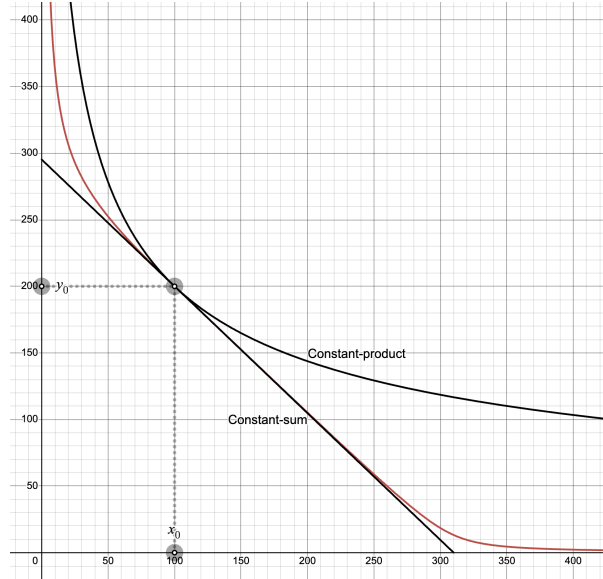


Figure 2: **EulerSwap AMM curve.** The EulerSwap curve (red line) consists of two sides with separate reserve values  $x_0, y_0$  and concentration parameters  $c_x, c_y$ , allowing liquidity to be distributed asymmetrically. This means liquidity can be more or less dense or concentrated on one side of the AMM relative to the other. The exchange rate at equilibrium is determined by the pricing parameters  $p_x, p_y$  and is fully flexible. You can interact with the curve [here](#) on Desmos to compare its behavior with traditional constant-sum and constant-product curves (black lines).

## 4 Conclusion

EulerSwap enhances Automated Market Making by leveraging just-in-time borrowing to expand liquidity depth dynamically. By integrating Euler's lending vaults, it enables market makers to amplify

positions efficiently while offering a customizable AMM curve that supports concentrated, distributed, and asymmetric liquidity structures. These features make EulerSwap adaptable to a wide range of trading strategies. However, this efficiency is not without some trade-offs. Swap accounts incur interest costs on borrowed assets, which can erode profitability if not offset by swap fees, interest on collateral, or token incentives. Additionally, EulerSwap inherits Euler's risk parameters, making it most effective for correlated asset pairs with high loan-to-value (LTV) ratios, while volatile assets pose higher liquidation risks if positions become imbalanced. Despite these trade-offs, EulerSwap represents a breakthrough in AMM design by repurposing idle liquidity in lending protocols, reducing capital inefficiencies, and optimizing swap execution. By catering to professional market makers, DAOs, and algorithmic traders, it positions itself as a powerful tool for on-chain liquidity optimization.

## Acknowledgments

During a security review of EulerSwap, Chris Michel noted similarities between some of its underlying concepts and BlackHoleSwap, a project prototype he had reviewed years earlier. While EulerSwap was designed independently and without influence from that project, the resemblance is significant enough to warrant its recognition here as prior art.

## 5 Appendix

### 5.1 Curve derivation

We start with an AMM containing reserves (initial liquidity) of two assets,  $x_0$  and  $y_0$ , and define two standard AMM curve models.

#### 5.1.1 Constant-sum and constant-product AMM curves

The constant-sum and constant-product AMM curves are given by:

$$x + y = x_0 + y_0, \quad (5)$$

$$xy = x_0 y_0. \quad (6)$$

These curves intersect at the point  $(x_0, y_0)$ , where their slope is:

$$\left. \frac{dy}{dx} \right|_{(x_0, y_0)} = -1.$$

To introduce arbitrary pricing, we extend these curves by adding pricing parameters  $p_x$  and  $p_y$ .

#### 5.1.2 Generalising to arbitrary pricing

We define new AMM equations incorporating custom pricing:

$$p_x x + p_y y = p_x x_0 + p_y y_0, \quad (7)$$

$$x^{p_y y_0} y^{p_x x_0} = x_0^{p_y y_0} y_0^{p_x x_0}. \quad (8)$$

Taking the derivatives of these equations with respect to  $x$ , we obtain:

$$\frac{dy}{dx} = -\frac{p_x}{p_y}, \quad (9)$$

$$\frac{dy}{dx} = -\frac{p_x}{p_y} \frac{x_0 y_0}{y_0 x}. \quad (10)$$

These results confirm that at equilibrium  $(x_0, y_0)$ , the slope is:

$$\left. \frac{dy}{dx} \right|_{(x_0, y_0)} = -\frac{p_x}{p_y}.$$

However, equation (8) involves an exponential form, which makes it difficult to evaluate numerically. To resolve this, we introduce the concept of virtual reserves.

#### 5.1.3 Introducing virtual reserves for simplicity

Note that in the interval  $0 < x < x_0$  trading should not depend on the initial amount of  $y_0$  liquidity. Swaps from should only increase liquidity beyond  $y_0$  and deplete the  $x_0$  liquidity. This suggests that we can split the domain of the AMM curves into two, and replace the real reserve  $y_0$  in the interval  $0 < x < x_0$  with an idealised virtual reserve  $y_v$ . Using the equilibrium condition:

$$p_x x_0 = p_y y_0,$$

we derive:

$$y_v = x_0 \frac{p_x}{p_y}.$$

Substituting into our pricing equations and simplifying, we obtain two modified AMM equations:

$$y = \frac{p_x}{p_y}(2x_0 - x), \quad (11)$$

$$y = \frac{p_x}{p_y} \frac{x_0^2}{x}. \quad (12)$$

Since these curves no longer pass through  $(x_0, y_0)$ , we correct them by adding back the difference  $y_0 - p_x/p_y x_0$ , leading to:

$$y = y_0 + \frac{p_x}{p_y}(x_0 - x), \quad (13)$$

$$y = y_0 + \frac{p_x}{p_y}(x_0 - x) \left( \frac{x_0}{x} \right). \quad (14)$$

#### 5.1.4 Unifying into a single curve

To create a single unified curve, we introduce a liquidity concentration parameter  $c_x$ , which determines the curve's behavior:

- When  $c_x = 1$ , the AMM functions as a constant-sum model.
- When  $c_x = 0$ , the AMM behaves as a standard constant-product AMM.
- Intermediate values of  $c_x$  create a hybrid liquidity curve.

This gives the final equation:

$$y = y_0 + \frac{p_x}{p_y}(x_0 - x) \left( c_x + (1 - c_x) \left( \frac{x_0}{x} \right) \right). \quad (15)$$

This is equivalent to the function  $f_1(x)$  given by equation (3) in the main text.

#### 5.1.5 Extending the curve to the $x > x_0$ region

For  $x > x_0$ , we construct a reflected curve that also passes through  $(x_0, y_0)$  and maintains the required derivative:

$$x = x_0 + \frac{p_y}{p_x}(y_0 - y) \left( c_y + (1 - c_y) \left( \frac{y_0}{y} \right) \right). \quad (16)$$

Since this equation defines  $x$  in terms of  $y$ , we invert it by solving for  $y$ , leading to the quadratic equation:

$$y = c_y y^2 + \left( \frac{p_x}{p_y}(x - x_0) - y_0(2c_y - 1) \right) y - (1 - c_y)y_0^2. \quad (17)$$

Taking only the positive real root, we obtain:

$$y = \frac{\sqrt{\left( \frac{p_x}{p_y}(x - x_0) + y_0(1 - 2c_y) \right)^2 + 4c_y(1 - c_y)y_0^2} - \left( \frac{p_x}{p_y}(x - x_0) + y_0(1 - 2c_y) \right)}{2c_y}. \quad (18)$$

This corresponds to the function  $f_2(x)$  given by equation (4) in the main text.

## 5.2 Invariant derivation

In traditional AMM protocols, the curve is typically defined as an implicit function of  $y$ . For example, the classic Uniswap AMM follows a constant-product equation:

$$xy = x_0y_0 \quad (19)$$

where  $x_0$  and  $y_0$  are the initial liquidity reserves. This equation defines an invariant condition, ensuring that any valid swap must satisfy:

$$xy \geq x_0y_0. \quad (20)$$

This condition guarantees that after any trade, the product of the new reserves remains at least as large as the initial product, ensuring that swaps cannot drain liquidity from the pool.

Rearranging this condition, we can express the invariant as a lower bound for  $y$ :

$$y \geq \frac{x_0y_0}{x}. \quad (21)$$

This means that for any valid trade, the resulting reserve state must lie on or above the AMM curve.

### 5.2.1 Extending the invariant to EulerSwap

For EulerSwap, we apply a similar principle. Given any reserve state  $(x, y)$ , we check whether it satisfies an equivalent invariant condition.

From equation (15), for values where  $x \leq x_0$ , the AMM curve constraint requires:

$$y \geq y_0 + \frac{p_x}{p_y} (x_0 - x) \left( c_x + (1 - c_x) \left( \frac{x_0}{x} \right) \right). \quad (22)$$

For values where  $x > x_0$ , we could derive a similar condition for  $y$  from equation (18). However, a simpler and computationally cheaper approach is to use the inverse equation (16), which defines the lower bound for  $x$  instead:

$$x \geq x_0 + \frac{p_y}{p_x} (y_0 - y) \left( c_y + (1 - c_y) \left( \frac{y_0}{y} \right) \right). \quad (23)$$

These conditions together define the valid liquidity states in EulerSwap, ensuring that the AMM remains balanced while allowing for greater flexibility in liquidity provisioning.

## 6 Disclaimer

This paper is for general information purposes only. It does not constitute investment advice or a recommendation or solicitation to buy or sell any investment and should not be used in the evaluation of the merits of making any investment decision. It should not be relied upon for accounting, legal or tax advice or investment recommendations. This paper reflects current opinions of the authors and is not made on behalf of Euler Labs or its affiliates and does not necessarily reflect the opinions of Euler Labs, its affiliates or individuals associated with Euler Labs. The opinions reflected herein are subject to change without being updated.