EulerSwap White Paper

Euler Labs

February 2025

Abstract

EulerSwap is an automated market maker (AMM) that integrates with Euler credit vaults to provide deeper liquidity for swaps. When a user initiates a swap, a smart contract called an EulerSwap operator borrows the required output token using the input token as collateral. This model enables up to 40x the liquidity depth of traditional AMMs by making idle assets in Euler more efficient. Unlike traditional AMMs, which often fragment liquidity across multiple pools, EulerSwap further increases capital efficiency by allowing a single, cross-collateralised credit vault to support multiple asset pairs at once. At its core, EulerSwap uses a flexible AMM curve to optimise swap pricing, ensuring deep liquidity while maintaining market balance. By combining just-in-time liquidity, shared liquidity across pools, and customisable AMM mechanics, EulerSwap reduces inefficiencies in liquidity provision, offering deeper markets, lower costs, and greater control for liquidity providers.

1 Introduction

EulerSwap is an automated market maker (AMM) that leverages Euler credit vaults for the provision of just-in-time liquidity as a way to provide deeper liquidity for swaps. EulerSwap differs from traditional AMMs in that it does not depend on many small liquidity providers (LPs) pooling their liquidity inside a single contract. Instead, each EulerSwap instance has a single LP − an account holder on Euler − whose deposits function as both liquidity for swaps and as margin collateral for building leveraged positions inside Euler.

When a user initiates a swap, EulerSwap dynamically borrows the required 'out token' using the 'in token' and the LP's initial margin collateral as backing. This borrowing approach ensures liquidity is provided just-in-time, only when needed, maximising capital efficiency. By borrowing liquidity to service swaps in this way, EulerSwap can turn $1M initial liquidity into the equivalent of a $40m deep liquidity pool on a traditional AMM (see Example below).

The way this works under the hood is by taking advantage of the 'operator' functionality of the Ethereum Vault Connector (EVC), which provides a way for Euler credit vault users to delegate authority to someone else to manage their account balances. An EulerSwap operator is a smart contract that manages an LP's account on their behalf using the logic encoded in a custom-built AMM smart contract. Specifically, an EulerSwap operator is able to rebalance the collateral and debt inside an LPs account in order to service swaps and earn swap fees.

Unlike conventional AMMs, which usually require separate liquidity pools for each asset pair, EulerSwap further increases capital efficiency by enabling the use of a single, cross-collateralised credit vault to support multiple asset pairs at once. This liquidity-sharing model ensures that idle liquidity can be used efficiently across multiple trading pairs simultaneously. This design resembles Curve's 3pool concept, but extends it to any number of asset pairings. Unlike traditional AMMs that silo liquidity into separate pools, EulerSwap's cross-collateralisation allows a single USDC credit vault to back multiple trading pairs simultaneously, maximising liquidity utilisation. A single USDC credit vault could be used to support swaps with USDT, USDE, DAI, and many other pairs all in the same block.

EulerSwap introduces a flexible AMM curve (illustrated in Figure 2) that supports deep liquidity for short-term swaps while maintaining a neutral net position over longer periods. A highly customisable curve allows for bespoke pricing strategies, asymmetric liquidity deposits, and single-sided liquidity concentration—all fully controlled by the LP. LPs can modify their EulerSwap parameters at any time by simply swapping out the operator smart contract in their Euler account for another one.

For end-users, EulerSwap offers a seamless, Uniswap-style experience, while behind the scenes, it harnesses advanced features such as dynamic liquidity borrowing, custom pricing curves, and shared liquidity provisioning. Liquidity providers, including professional market makers, token issuers, and DAOs, gain unprecedented capital efficiency. Meanwhile, swappers—ranging from leverage traders and aggregators to MEV bots—benefit from deeper liquidity and optimised trade execution.

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 an account on Euler with 1M USDC deposited as initial liquidity. Using maximum leverage your account could hypothetically 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.

To enable swaps and earn additional yield, you install an EulerSwap operator on your account to facilitate swaps. Now let's say another user wants to swap 10M USDC for USDT. The steps are as follows:

1. The swapper sends 10M USDC to your EulerSwap operator as the swap input amount.
2. The operator deposits the 10M USDC as collateral in Euler.
3. The operator then borrows approximately 10M USDT against the account's collateral, which includes your original deposit, alongside the swap input.
4. The operator sends the borrowed USDT to the user as the swap output.

Importantly, this isn't a 1:1 swap because: a) the EulerSwap operator charges a fee for facilitating the swap; and b) the exact swap output is determined by an AMM curve inside the operator, chosen by you, which factors in increasingly large price impact as the swap input amount increases. After the swap, your Euler account now holds 11M USDC deposits and 10M USDT debt. 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 inside the operator helps to ensure that imbalances in collateral and debt on the account encourage swaps that bring your account back to neutrality. The AMM is designed to help ensure that positions on your account do not remain open for extended periods, reducing your account's exposure to borrowing costs. Over time, your account will incur costs due to small interest rate differentials, whilst generating significant swap fees through utilisation of idle liquidity in Euler's credit vaults. The whole process is depicted in Fig. 1.

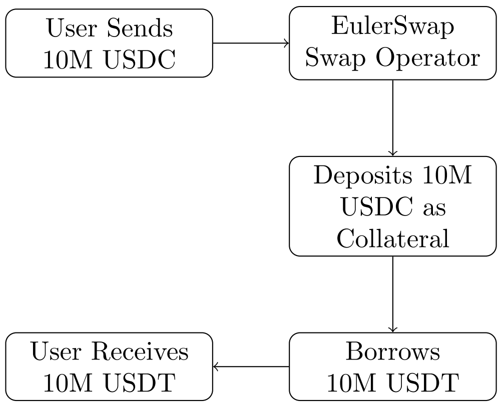

Figure 1: **Swap flow in EulerSwap AMM**. EulerSwap's just-in-time liquidity borrowing. The swap operator dynamically increases liquidity by borrowing the "out token" against the "in token."

3 Virtual reserves and debt limits

Since EulerSwap AMMs do not hold the assets used to service swaps at all times, they do swap calculations based on 'virtual' reserves and debt limits, rather than on 'real' reserves.

Each EulerSwap LP can configure independent virtual reserve levels. These reserves define the maximum debt exposure an AMM will take on. For instance, if a user deposits \$1,000 in collateral and sets virtual reserves at \$5,000 per vault, the AMM effectively supports up to \$10,000 in combined swap depth, with a loan-to-value (LTV) ratio of 83.3%.

Note that the effective LTV must always remain below the borrowing LTV of the credit vault to prevent liquidation. Additionally, different AMM curves influence whether the maximum virtual reserves are achievable.

4 Curve

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

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

Unlike most AMM curves, which are usually defined by a single convex function, EulerSwap uses a piecewise-defined curve, with different functions guiding trading behaviour either side of the equilibrium point. In the domain $0 < x \leq x_0$, the curve is defined by

$$y = y_0 + \frac{p_x}{p_y}(x_0 - x)\left(c_x + (1 - c_x)\left(\frac{x_0}{x}\right)\right), \tag{2}$$

with y depending x. In the region $x_0 < x$, however, we let x become the dependent variable, so that the domain is $0 < y < y_0$, and the curve is defined by

$$x = x_0 + \frac{p_y}{p_x}(y_0 - y)\left(c_y + (1 - c_y)\left(\frac{y_0}{y}\right)\right), \tag{3}$$

with x depending on y. Although defining an AMM using this technique is unusual, both curves have inverses, allowing trading behaviour to be fully defined for any swap input or output amount in any direction.

The c_x, c_y parameters in the equations are liquidity concentration parameters that control how liquidity is distributed along the curve, with values closer to 1 concentrating liquidity around equilibrium and values closer to 0 distributing it across a wider price range. This flexibility enables EulerSwap to be used for entirely new use cases or to simulate the behaviour of atypical AMM protocols, such as the MakerDAO Peg Stability Module (PSM). A full derivation is given in the Appendix 6.

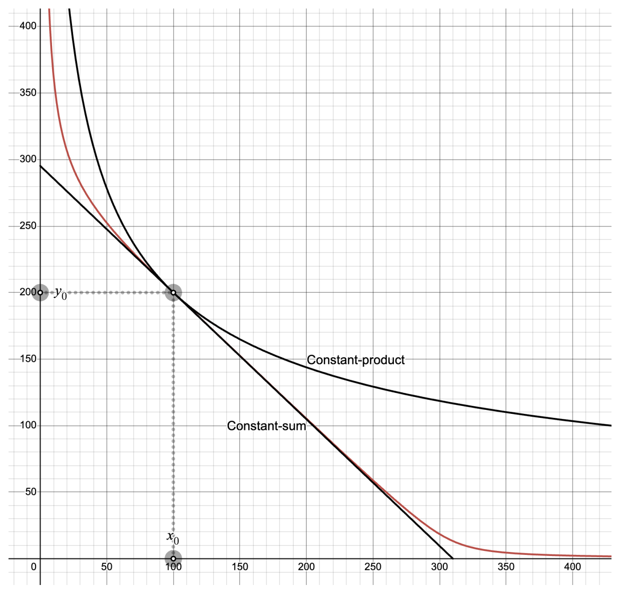

Figure 2: **EulerSwap AMM curve.** The EulerSwap curve (red line) consists of two sides with separate reserve values x_0, y_0 and liquidity concentration parameters c_x, c_y, allowing liquidity to be distributed asymmetrically. This means liquidity can be more or less dense or concentrated on one side 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 behaviour with traditional constant-sum and constant-product curves (black lines).

5 Conclusion

EulerSwap enhances AMMs by leveraging just-in-time liquidity to expand the depth of liquidity available to swappers. By integrating Euler's credit vaults, it enables LPs to earn swap fees on top of their ordinary deposits while offering a customisable 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 operators 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 optimising swap execution. By catering to professional market makers, DAOs, and algorithmic traders, it positions itself as a powerful tool for on-chain liquidity optimisation. Future research could explore optimal strategies for setting pricing and liquidity concentration parameters dynamically based on market conditions. EulerSwap's design opens new possibilities for AMM efficiency, setting a foundation for further DeFi innovation.

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.

6 Appendix

6.1 Curve derivation

We begin with an Automated Market Maker (AMM) holding initial liquidity reserves of two assets, X and Y, denoted as x_0 and y_0, respectively. In the absence of trading, the AMM remains at equilibrium at the point (x_0, y_0).

Our goal is to derive a curve for a constant-function trading market maker (CFMM) that supports swaps between the two assets with the following properties:

- Passes through the equilibrium point (x_0, y_0).
- Maintains an exchange rate, given by the slope of the AMM curve, of $-p_x/p_y$ at (x_0, y_0).
- Allows liquidity concentration to be adjusted via parameters c_x and c_y, which control the liquidity available for swaps to the left and right of the equilibrium point.

To develop such a function, we first introduce two fundamental AMM curve models.

6.1.1 Constant-sum and constant-product AMM curves

The canonical constant-sum (CSMM) and constant-product (CPMM) market making curves are given by:

$$x + y = x_0 + y_0, \tag{4}$$
$$xy = x_0 y_0. \tag{5}$$

The CSMM is simply a line, whilst the CPMM is a hyperbola. These curves can be thought of as two extremes when it comes to the distribution of liquidity. The CSMM concentrates liquidity at a single exchange rate, whilst the CPMM distributes liquidity across a wide range of different exchange rates. By default, these curves intersect at the equilbirium point (x_0, y_0), where their slopes are:

$$\frac{dy}{dx} = -1, \tag{6}$$
$$\frac{dy}{dx} = -\frac{y}{x}. \tag{7}$$

Since real-world markets often operate at variable exchange rates at equilibrium, we introduce custom pricing parameters p_x and p_y to allow flexibility in defining the slope at the equilibrium point:

$$p_x x + p_y y = p_x x_0 + p_y y_0, \tag{8}$$
$$x^{p_y y_0} y^{p_x x_0} = x_0^{p_y y_0} y_0^{p_x x_0}. \tag{9}$$

The CSMM is now a line with a slope parameterised by the ratio of the pricing parameters, whilst the CPMM is now a *weighted* hyperbola. Taking the derivatives of these equations with respect to x, we obtain:

$$\frac{dy}{dx} = -\frac{p_x}{p_y}, \tag{10}$$
$$\frac{dy}{dx} = -\frac{p_x}{p_y}\frac{x_0 y}{y_0 x}. \tag{11}$$

These results confirm that at equilibrium (x_0, y_0), the slope of both functions is:

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

Whilst these curves sufficiently generalise the constant-sum and constant-product curves to an arbitrary price at equilibrium, in practice, the modified CPMM equation (9) has limited practical application as a CFMM because it involves an exponential form that is computationally intensive and impractical for on-chain calculations. We therefore seek a simpler equation that preserves the spirit of the trading behaviour of the generalised CPMM; that is, some kind of herpbola that passes through the equilibrium point (x_0, y_0) and maintains an exchange rate of $-p_x/p_y$ at that point. To resolve this, we introduce the concept of artificial reserves.

6.1.2 Introducing artificial reserves to create a new, simpler curve

Note that in the interval $0 < x < x_0$ swaps should only increase liquidity beyond y_0 and deplete x_0 liquidity. That is, our trading function in this interval need not depend on the initial amount of y_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 carefully chosen artificial reserve y_v designed to eliminate the exponential form in the weighted hyperbola.

Re-arranging equations (8) and (9) into explicit functions of y, we obtain

$$y = y_0 + \frac{p_x}{p_y}(x_0 - x), \tag{12}$$

$$y = y_0 \left(\frac{x_0}{x}\right)^{\frac{p_y y_0}{p_x x_0}}. \tag{13}$$

In this view, it is easy to see that a substitution of $y_0 \to y_v$, given by

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

will eliminate the exponential form in equation (13). We then have equations

$$y = \frac{p_x}{p_y}(2x_0 - x), \tag{14}$$

$$y = \frac{p_x}{p_y}\frac{x_0^2}{x}. \tag{15}$$

Note that simply shifting the curve up or down does not impact the slope or shapes of the curves and therefore has no impact on trading behaviour. Since these curves no longer pass through (x_0, y_0), we therefore correct them by adding back the difference $y_0 - p_x/p_y x_0$. This leads to:

$$y = y_0 + \frac{p_x}{p_y}(x_0 - x), \tag{16}$$

$$y = y_0 + \frac{p_x}{p_y}(x_0 - x)\left(\frac{x_0}{x}\right). \tag{17}$$

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

$$\frac{dy}{dx} = -\frac{p_x}{p_y}, \tag{18}$$

$$\frac{dy}{dx} = -\frac{p_x}{p_y}\left(\frac{x_0}{x} + \frac{x_0(x_0 - x)}{x^2}\right). \tag{19}$$

These results confirm that at equilibrium (x_0, y_0), the slope of both functions is:

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

6.1.3 Unifying into a single curve in the region $0 < x < x_0$

To create a single unified curve, we introduce a liquidity concentration parameter $c_x \in [0, 1]$ that determines the curve's behaviour:

- When $c_x = 1$, the AMM functions as a constant-sum AMM.
- When $c_x = 0$, the AMM behaves as a constant-product-like AMM.
- Intermediate values of c_x create a hybrid trading function, with liquidity that is more or less concentrated around the equilibrium point.

This parameterisation leads to the following equation:

$$y = y_0 + \frac{p_x}{p_y}(x_0 - x)\left(c_x + (1 - c_x)\left(\frac{x_0}{x}\right)\right). \tag{20}$$

This equation is equivalent to equation (2) in the main text. It is implemented as the function 'f()' in Solidity within the 'EulerSwapCore.sol' contract. It serves two main purposes:

- Swap quoting: Providing quotes for y given x in the domain $0 < x < x_0$.
- System invariant: Acting as a key invariant within the EulerSwap system (detailed below).

To function as a complete trading function, equation (20) can also be inverted to compute x given y in the range $0 < x < x_0$. By rearranging the formula into a quadratic in x, we derive:

$$c_x x^2 + \left(\frac{p_y}{p_x}(y - y_0) - (2c_x - 1)\, x_0\right) x - (1 - c_x)\, x_0^2 = 0. \tag{21}$$

Taking only the positive real root, we solve for x as:

$$x = \frac{\sqrt{\left(\frac{p_y}{p_x}(y - y_0) - (2c_x - 1)\, x_0\right)^2 + 4c_x(1 - c_x)\, x_0^2} - \left(\frac{p_y}{p_x}(y - y_0) - (2c_x - 1)\, x_0\right)}{2c_x}. \tag{22}$$

This inverse function is implemented as 'fInverse()' in the 'EulerSwapPeriphery.sol' contract. While it could theoretically be used as an invariant, this is not necessary. Moreover, the equation is less computationally efficient than equation (20), which is why it is only used in the periphery of EulerSwap for quoting purposes.

6.1.4 Extending the curve to the $x > x_0$ region

It is important to remember that equation (20) only works as a trading function in the domain $0 < x < x_0$. For $x > x_0$, observe that what we want is a curve very similar in nature to equation (20), except one that fulfills similar trading behaviour for asset Y in its equivalent domain $0 < y < y_0$. If we simply reflect equation (20), by swapping $y \to x$ and $x \to y$ we obtain

$$x = y_0 + \frac{p_x}{p_y}(x_0 - y)\left(c_x + (1 - c_x)\left(\frac{x_0}{y}\right)\right). \tag{23}$$

This is something close to what is needed, except that it:

- Passes through the equilibrium point (y_0, x_0).
- Maintains an exchange rate, given by the slope of the AMM curve, of $-p_y/p_x$ at (y_0, x_0).
- Has liquidity concentration parameter c_x even though it needs to control the concentration of Y liquidity.

We therefore also swap the parameters of the curve around as well to obtain a new curve

$$x = x_0 + \frac{p_y}{p_x}(y_0 - y)\left(c_y + (1 - c_y)\left(\frac{y_0}{y}\right)\right). \tag{24}$$

This equation is equivalent to equation (3) in the main text. It provides an equation to solve for swaps in which x is given and y is unknown when we are in the domain $0 < y < y_0$ (equivalent to $x > x_0$). Note that it is not really a new curve, but rather simply a re-parameterisation of equation (20). It is therefore also implemented as the function 'f()' in Solidity within the 'EulerSwapCore.sol' contract.

The inverse of equation (24) is similarly obtained by re-parameterising equation (22) to get:

$$y = \frac{\sqrt{\left(\frac{p_x}{p_y}(x - x_0) - (2c_y - 1)\, y_0\right)^2 + 4c_y (1 - c_y)\, y_0^2} - \left(\frac{p_x}{p_y}(x - x_0) - (2c_y - 1)\, y_0\right)}{2c_y}. \tag{25}$$

This inverse function is also implemented as 'fInverse()' in the 'EulerSwapPeriphery.sol' contract.

6.2 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:

$$xy = x_0 y_0 \tag{26}$$

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

$$xy \geq x_0 y_0. \tag{27}$$

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:

$$y \geq \frac{x_0 y_0}{x}. \tag{28}$$

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

6.2.1 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 (20), for values where $0 < x < x_0$, the AMM curve constraint requires a lower bound for y:

$$y \geq y_0 + \frac{p_x}{p_y}(x_0 - x)\left(c_x + (1 - c_x)\left(\frac{x_0}{x}\right)\right). \tag{29}$$

For values where $0 < x < x_0$, which is equivalent to $x > x_0$, we simply use a lower bound for x instead:

$$x \geq x_0 + \frac{p_y}{p_x}(y_0 - y)\left(c_y + (1 - c_y)\left(\frac{y_0}{y}\right)\right). \tag{30}$$

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

7 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.