New Proteus Explainer

April 2022

Abstract

Since the release of the Proteus white paper, we have made some changes to the algorithm. The high-level design goal is still the same: create a generalized algorithm that can replicate any bonding curve. However, instead of using one complicated equation for the bonding curve, we use a series of simpler equations, with each equation applicable to a different radial slice of the curve. These changes make the algorithm more flexible and computationally efficient. Before going through this explainer, we recommend reading the original white paper, especially Sections 5, 6, 7 and Appendixes A, B.

1 Introduction

In our original Proteus white paper [link], we described how we can take an "identity curve" and use it as a bonding curve for an AMM. In this algorithm, rather than scaling the curve to the pool's balances (hard), we scale the balances to the identity curve (easy). Section 6 in the original white paper explains how this is accomplished. We can use any equation as a bonding curve as long as we can compute the intersection of the equation and the line $y/x = m$, where m is the ratio between the AMM's x balance and y balance (see Figure 1).

In the white paper and our initial Solidity implementation, we used a generalized conic section to parameterize our identity curve:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

This implementation had many drawbacks, namely conics are difficult to parameterize for this use case. Hence, we created a new identity curve implementation based on a modified constant product:

$$(x + a)(y + b) = 1$$

This paper will explain how this new implementation works. It will assume prior knowledge of the original white paper, especially Sections 5, 6, 7, and Appedixes A, B.

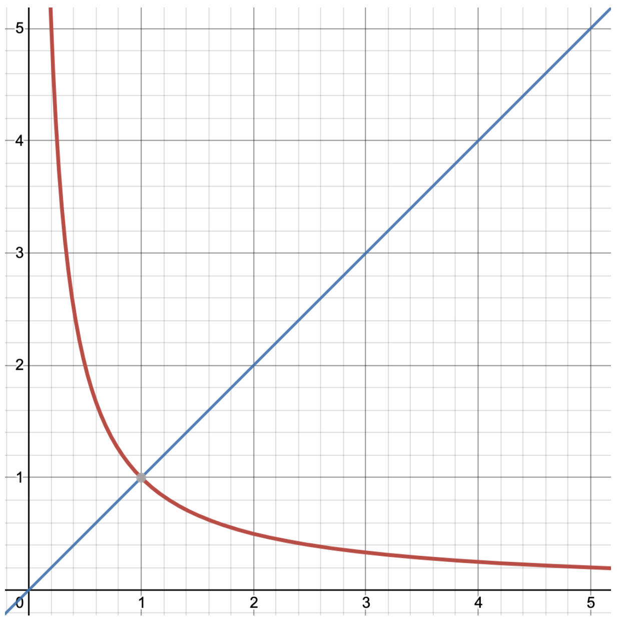

Figure 1: The radial line (blue) represents the ratio between the x and y balances. The curve (red) represents the equation for the AMM's identity curve. As long as we can compute this intersection, we can scale the balances to the identity curve using the Proteus algorithm.

2 Using p and m to fit a series of identity curves

As demonstrated in the original white paper, Section 5, the instantaneous price ($p = -\frac{dy}{dx}$) of the curve is the same along the radial line $y/x = m$ for all positive values of m, regardless of how many tokens are held by the AMM (see Figure 2).

Therefore, we will parameterize our identity curve based on the ratio, m_i, and the price, p_i, at that ratio.

$$\mathbf{m} = [m_1, m_2, ..., m_i, m_{i+1}, ..., m_n]$$

$$\mathbf{p} = [p_1, p_2, ..., p_i, p_{i+1}, ..., p_n]$$

With these parameters, we have divided the AMM into $n + 1$ slices. Rather than fit one identity curve to all of these parameters, we will generate $n + 1$ identity curves that fit the parameters in their respective slices. The ending instantaneous price of each slice is the same as the starting instantaneous price of the proceeding slice. The exception being the slices adjacent to the x and y intercepts, which will have asymptotes. Therefore, there will not be any discontinuity in the instantaneous price.

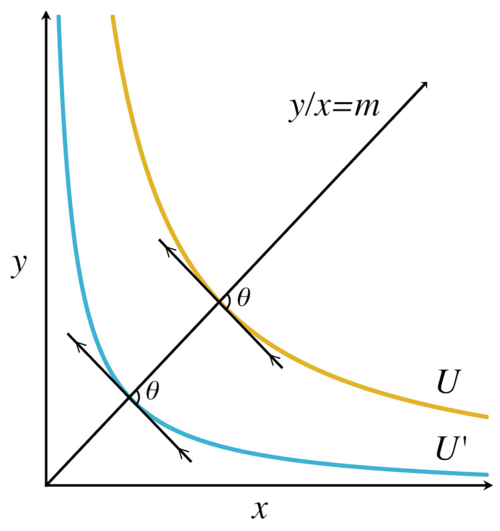

Figure 2: Where U and $U^{'}$ intersect the line $y/x = m$, the instantaneous prices are the same because their tangent lines are parallel.

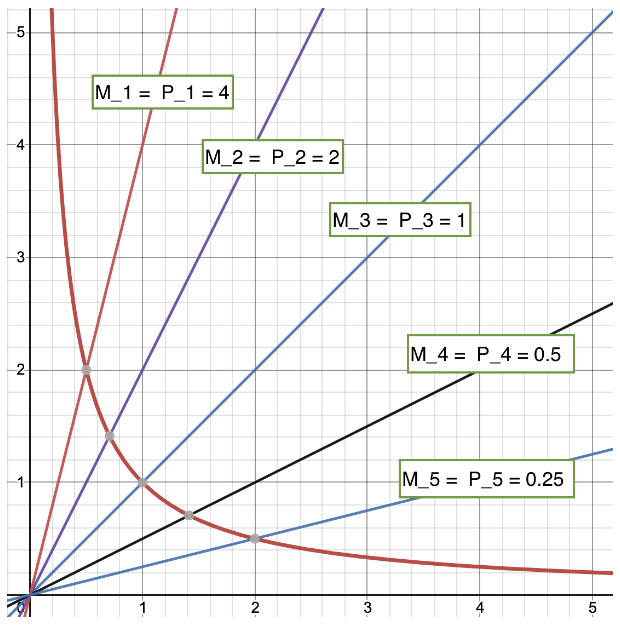

Figure 3: Each radial line is associated with a ratio, m, and an instantaneous price, p_i. For every bonding curve, the relationship between m and p is invariant. This figure shows the relationship between m and p for constant product, $xy = k$, where the ratio is the same as the instantaneous price.

3 Fitting an identity curve to each slice

As stated previously, the identity curve for each slice will be a modified constant product with a and b being the two free parameters. We set $k = 1$ for the identity curve because in our case, the scale of the curve does not matter, only the relationship between the instantaneous price (p) and the balance ratio (m).

For any given slice, we know the associated $m_i, m_{i+1}, p_i, p_{i+1}$. With this information, we can find a and b by solving the following system of equations:

$$\frac{\sqrt{p_i} + b}{\frac{1}{\sqrt{p_i}} + a} = m_i$$

$$\frac{\sqrt{p_{i+1}} + b}{\frac{1}{\sqrt{p_{i+1}}} + a} = m_{i+1}$$

To get an intuition how these equations work, consider that for regular constant product, where $xy = 1$, the instantaneous price is the same as the ratio between x and y:

$$\frac{y}{x} = p$$

If we know the instantaneous price, p_i, then we can easily calculate the x or y value on the $xy = 1$ curve:

$$x = \sqrt{p}$$

$$y = \frac{1}{\sqrt{p}}$$

All we need to do now is find the a and b values that shift this point such that the ratio between x and y is now m_i. Figure 4 shows graphically what is happening.

As mentioned previously, there are two additional slices adjacent to the y and x axis. We use a slightly different approach to finding the identity curve for these two sections. For brevity, we will explain how to fit the y axis curve; however, the approach is easy to generalize for the x axis curve. In this case, we only need to consider one free parameter, b:

$$x(y + b) = 1$$

We can find the value of b by solving the following equation:

$$\sqrt{p_1} + b = \frac{m_1}{\sqrt{p_1}}$$

Where p_1 and m_1 are respectively the starting instantaneous price and starting ratio for the first slice. For the x intercept, we follow the same steps, except we use p_n and m_n.

4 Visualizing the identity curves

In this section, we provide a graphical representation of Proteus' identity curves for the sake of visualization. In Figure 4, we demonstrate how the identity curve for a given slice is created. In Figure 5 we demonstrate what a series of identity curves for concentrated liquidity would look like.

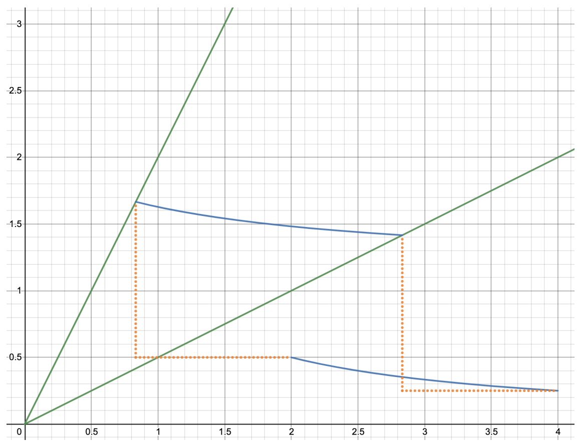

Figure 4: We are given the parameters $\mathbf{m} = [2, 0.5]$ and $\mathbf{p} = [0.25, 0.0625]$. We shift over a section of the $xy = 1$ curve (blue) so that it fits within the slice (green). The the horizontal shift is $a = \frac{7}{6}$ and the vertical shift is $b = -\frac{7}{6}$ (dashed orange). You can view the Desmos file for the graph at this link.

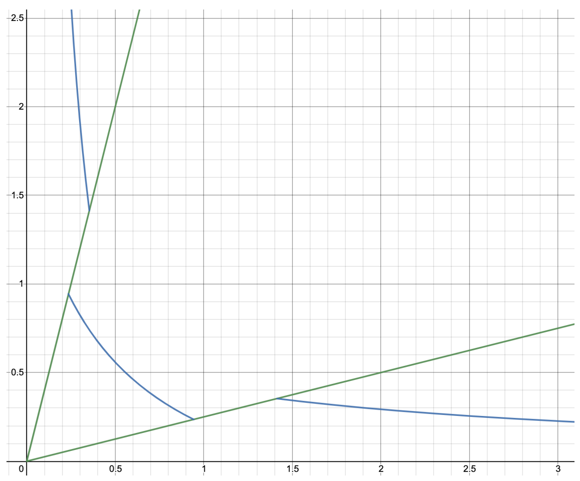

Figure 5: We are given the parameters $\mathbf{m} = [4, 0.25]$ and $\mathbf{p} = [2, 0.5]$. We then fit our identity curves (blue) to each slice (green). In the middle slice, $a = b = -0.4714$; and in the peripheral slices, $a = b = 1.414$. Note how the center curve is closer to the origin. That means liquidity over that slice is more highly concentrated than the peripheral slices. You can view the Desmos file for the graph at this link.

5 Computing swaps

Other than the identity curve, the main difference between this new version of Proteus and the version in the original white paper is that in this new version we may need to cross between multiple slices during a transaction. For swaps that start and end in the same slice, we can use the same high level algorithm described in the white paper, Section 7. For transactions that span multiple slices, we need to iterate through each slice, starting at the current slice and moving in the direction of the swap. If the user is swapping Δx for Δy, then we will iterate through slices in a clockwise direction. For swaps exchanging Δy for Δx, then we iterate in the counter-clockwise direction.

To streamline the process, for each slice, m_i, we pre-compute the x and y values of the identity curve where it crosses the radial lines. This allows us to easily check to see if a given swap will exceed the current slice or not. All we need do is re-scale the balances to the new identity curve in each slice, then check if the scaled swap amount is greater than at the end of the slice. If yes, then proceed to the next slice and repeat the process. If no, then compute the final balances in that slice.

6 Computing deposits (and withdrawals)

There are two types of deposit (or withdrawal). In the first type, the user specifies how many x tokens to add to the pool. The AMM then computes how many LP tokens to mint to the user.

In the second type, the user specifies how many LP tokens they want to mint and the pool then computes how many x tokens the user must deposit.

At a high level, the first type of deposit involves calculating the change in the AMM's "utility function" (see the original white paper, Appendix A). The utility function maps the AMM's balances to a scalar. This scalar is how the AMM measures the total value of the tokens it holds. In the case of Proteus, we use the scale factor between the identity curve and the AMM's balances as the scalar value (see white paper Section 6 regarding the scale factor, λ). The amount of tokens minted, ΔLP is proportional to the change in the utility function:

$$U(x, y) = \lambda$$

$$\Delta LP = \frac{U(x + \Delta x, y) - U(x, y)}{U(x, y)}(LP)$$

How do we calculate the percent change in the utility function for a deposit that traverses multiple slices? In each slice, we calculate the change in utility if we deposit up to the end of the slice. We then add the ΔLP minted to the total supply of LP tokens. We keep iterating through the slices, incrementing the total supply as we go, until reaching the end of the deposit.

The second type of deposit, where the user specifies how many LP tokens they want to mint, works similarly to the first except in reverse. The ΔLP specified by the user has an implied increase in the utility function of the AMM. In each slice, we can compute how much Δx needs to be added to increase the AMM's utility. As we iterate through the slices, we decrement from the ΔLP minted to the user and increment the total supply of LP tokens.

Withdrawals are effectively the same as deposits, except instead of adding and minting tokens, we subtract and burn tokens.