A study on AMMs for trading fixed yield and Pendle V2's Principal Token AMM

Vu Nguyen
vu@pendle.finance

October 14, 2022

Abstract

Fixed-yield products play a key role in any financial systems by providing certainty in a turbulent market. In DeFi, fixed-yield products allow users to speculate on fixed-yield by trading principal tokens (PT). PTs are analogous to zero-coupon bonds where it is redeemable for a fixed amount of assets after expiry. In this paper, we will first compare the efficiencies of current PT AMM models. Afterwards, we will introduce Pendle V2's PT AMM, which uses Notional Finance's AMM model [5] as a baseline.

1 A study on AMMs for trading fixed yield

1.1 Definitions

yield generating asset is an asset following the Generic Yield Generating Pool model, as described in the Standardized Yield paper [7] . The value of the yield generating asset is measured in **asset** (as defined in the GYGP model). E.g: cDAI (DAI deposited into Compound) is a **yield generating asset** and DAI is the corresponding **asset**.

expiry the end of the period where yield can be traded until

normalised time t is a measure of time left until the expiry, $0 \leq t \leq 1$, such that $t = 0$ at expiry

Principal token (PT) is a token that allows users to get back 1 **asset** worth after the expiry. As the final value of **principal token** is known, PT holders receive fixed yield on expiry. Buying PT is also equivalent to shorting yield, profiting in the short run if yield drops. E.g: 1 PT-DAI-1stJan2023 will be redeemable for the principal of 1 DAI after 1st Jan 2023.

principal token price at time t is $price_p(t)$, which is the price of **principal token** in terms of **asset**

asset price in principal at time t, $price_a(t)$ is simply $\frac{1}{price_p(t)}$, which is the price of the $asset$ in terms of **principal token**

implied interest rate is the annual average interest rate from now (t) until the expiry that the market is implying, by trading **principal token** at $price_p(t)$. In this paper, an interest rate of x% is represented by the number $1 + \frac{x}{100}$

Since holding 1 **asset** = $price_a(t)$ **principal tokens** now will return $price_a(t)$ **asset** on expiry, the interest from now to expiry is $price_a(t)$

As such, the annual interest rate (which is also the implied interest rate) is:

$$price_a(t)^{\frac{t_{oneYear}}{t}}$$

principal token AMM is an AMM that allows users to buy or sell principal tokens against the **asset**. Buying principal tokens will increase its price $price_p(t)$ and decrease **asset price in principal** $price_a(t)$, hence decreasing the **implied interest rate**. Conversely, selling principal tokens will increase the **implied interest rate**.

Let n_{asset} and n_{pt} be the amounts of assets and principals in the AMM pool. We will also use $x = n_{asset}$ and $y = n_{pt}$ as the short forms.

proportion p is a measure of the proportion of principal tokens in the pool:

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

It follows that:

$$\frac{p}{1-p} = \frac{y}{x} \tag{1}$$

1.2 Existing models for principal token AMM

1.2.1 Constant geometric mean for PT and asset

In this model, the AMM trades PT and asset based on a constant geometric mean formula (similar to Balancer's formula [3])

$$x^{w_x} \times y^{w_y} = k \tag{2}$$

$$w_x + w_y = 1 \tag{3}$$

This model is used in Pendle V1 (Sushiswap pool which follows Uniswap V2 formula [1] where $w_x = w_y = 0.5$) and Apwine [2] (where $w_x = w_y = 0.5$ at the start, and w_y increases over time)

asset price in principal is $-\frac{dy}{dx}$
From (2):

$$y = \frac{k^{\frac{1}{w_y}}}{x^{\frac{w_x}{w_y}}}$$

$$\frac{dy}{dx} = -\frac{w_x}{w_y} \times \frac{k^{\frac{1}{w_y}}}{x^{1+\frac{w_x}{w_y}}}$$

$$-\frac{dy}{dx} = \frac{w_x}{w_y} \times \frac{(x^{w_x} \times y^{w_y})^{\frac{1}{w_y}}}{x^{1+\frac{w_x}{w_y}}}$$

$$price_a = \frac{w_x}{w_y} \times \frac{y}{x}$$

From (1), we get:

$$price_a = \frac{w_x}{w_y} \times \frac{p}{1-p} \tag{4}$$

1.2.2 YieldSpace's constant power sum

In this model, the AMM trades PT against asset based on a constant power sum formula that changes over time:

$$x^{1-t} + y^{1-t} = k \tag{5}$$

This model is used in Yield Protocol [8], Element Finance [4] and Sense Finance [6].

Asset price in principal is $-\frac{dy}{dx}$
From (5):

$$y^{1-t} = k - x^{1-t}$$

Differentiate both sides w.r.t x:

$$(1-t) \times y^{-t} \times \frac{dy}{dx} = -(1-t) \times x^{-t}$$
$$-\frac{dy}{dx} = \left(\frac{y}{x}\right)^t$$
$$price_a = \left(\frac{p}{1-p}\right)^t$$

1.2.3 Notional AMM

The Notional AMM [5] is not defined based on a formula on the PT and asset reserves. Instead, it's defined by this formula for the **asset price in principal**:

$$price_a(t) = \frac{ln\left(\frac{p(t)}{1-p(t)}\right)}{rateScalar(t)} + rateAnchor(t) \tag{6}$$

rate scalar $rateScalar(t)$ at time t is a parameter used in calculating the price at time t, to adjust the capital efficiency.

$$rateScalar(t) = \frac{scalarRoot}{t}$$

rateAnchor(t) is a parameter used to adjust the interest rate around which the trading will be the most capital efficient, at time t.

$rateAnchor(t)$ is adjusted before every trade, such that the pre-trade implied interest rate is the same as the implied interest rate after the last trade.

scalarRoot is a fixed parameter for each AMM pool, to adjust the trade-off between the capital efficiency of the market and the trade-able range of interest. It is the value for $rateScalar$ at $t = 1$.

1.3 Framework for comparing principal token AMM models

1.3.1 A graphical representation of the AMM models

For all the three models, **asset price in principal** at a certain time t can be written as a function of **proportion** $f(p)$.

If we plot $f(p)$, we can see how the price (of asset/principal token) moves when users buy or sell principal tokens (hence, changing the proportion).

The y-axis represents the implied interest rate (that is not annualized). For example, a value of 1.13 represents 13% interest from now to expiry.

In Figure 1:

- black is the geometric mean model with $w_x = 0.5$
- blue is the YieldSpace model for $t = 0.3$

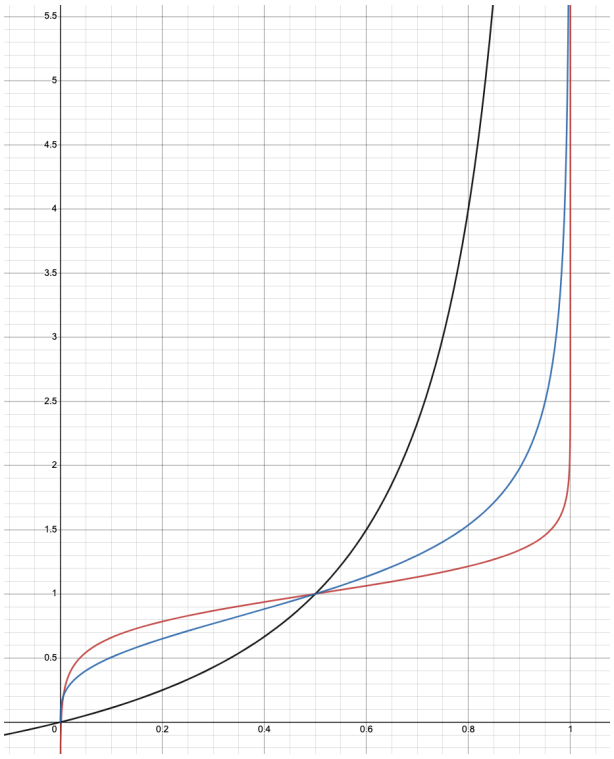

Figure 1: Price over proportion

- red is the Notional model with $anchorRate = 1$, $scalarRoot = 2$, $t = 0.3$

The slope of the curve represents how much price (and hence, implied interest rate) is moved by a certain change in proportion. Intuitively, a gentler slope means higher capital efficiency. From the graph alone, we can see that for the particular chosen set of parameters, the Notional AMM model is the most capital efficient.

1.3.2 Customisability

Both the constant geometric mean model and YieldSpace's constant power sum model are not customizable which means that the same curve is used for every yield generating asset.

There are two main parameters that can be set for each pool in the Notional AMM: the **scalar root** and the initial **rate anchor**.

In principle, higher customizability means that the curve for each Notional AMM's pool is more specialised in trading the yield of a particular yield generating asset.

In this paper, we will define a heuristic in setting **scalar root** and initial **rate anchor** in a Notional AMM pool, to maximise the capital efficiency for trading a particular yield generating asset:

Setting scalar root and initial rate anchor
To set the parameters, there are three assumptions about the yield generating asset:

- The market interest rate will be trading around a $rate_{expected}$ level. This is a guess of where the

average interest rate will be.

- The implied interest rate will not trade above $rate_{max}$.
- The implied interest rate will not trade below 1. This is always true since principal token price must not exceed asset price.

Let t_{start} be the start of the pool and $yearsLeft = \frac{t_{start}}{t_{oneYear}}$

From the first assumption, at the start of the pool, we will set the initial rate anchor to be $(rate_{expected})^{yearsLeft}$.

As observed from the curve for the Notional AMM model, which takes its shape from the logit curve, the slope becomes exceedingly high at the two extremes. We will define the **reasonable trading range** as the range from $p = 0.1$ to $p = 0.9$, where slippage is considered reasonable.

At the start of the pool (t_{start}), we can set the **scalar root** such that the initial reasonable trading range can cover implied interest rates from 1 to $rate_{max}$.

$$f(0.9) \geq (rate_{max})^{yearsLeft} \tag{7}$$

$$f(0.1) \leq 1 \tag{8}$$

From (7)

$$\frac{ln(\frac{0.9}{1-0.9})}{rateScalar} + rate_{expected} \geq (rate_{max})^{yearsLeft} \tag{9}$$

$$rateScalar \leq \frac{ln(9)}{(rate_{max})^{yearsLeft} - (rate_{expected})^{yearsLeft}} \tag{10}$$

From (8)

$$\frac{ln(\frac{0.1}{1-0.1})}{rateScalar} + rate_{expected} \leq 1 \tag{11}$$

$$rateScalar \leq \frac{ln(9)}{(rate_{expected})^{yearsLeft} - 1} \tag{12}$$

From (10), (12) and the fact that we would want a high $rateScalar$ to maximise capital efficiency, we can set $scalarRoot$ such that at t_{start}:

$$rateScalar(t_{start}) = min\left(\frac{ln(9)}{(rate_{max})^{yearsLeft} - (rate_{expected})^{yearsLeft}}, \frac{ln(9)}{(rate_{expected})^{yearsLeft} - 1}\right) \tag{13}$$

1.3.3 Capital efficiency

To compare capital efficiency, we can assume a fixed amount of liquidity in an AMM pool, and measure how much a user can trade, in terms of principal token amount, to move the implied rate by a certain amount.

Scenario 1
Simulating fixed yield trading for cUSDC (USDC deposited into Compound):

- The expiry is in 2 years. $t_{start} = 1$ and $t_{oneYear} = 0.5$

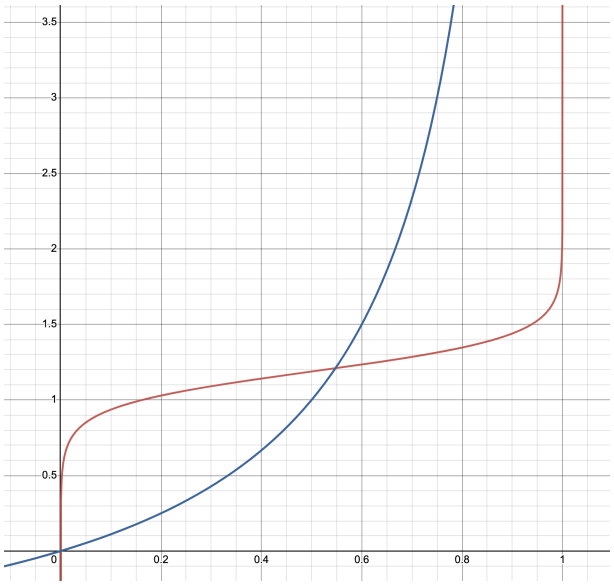

Figure 2: Scenario 1, t = 1

- Average interest rate $rate_{expected} = 1.09$ (9%)
- Max implied interest rate $rate_{max} = 1.2$ (20%)
- Assume the pool has 1 million USDC worth of PT and USDC

For the Notional AMM, following the heuristics defined, we get initial rate anchor = 1.1881, $rateScalar(t_{start}) = 8.7226$

At t=1, we can see the graphs of the 3 models in Figure 2 (Note that both the YieldSpace and geometric mean model curve overlaps, represented by the blue curve)

Assuming the initial interest rate is at 1.09 (price=1.1881), we can calculate how much PT we could sell to push the interest rate to 1.11 (price=1.221):

- For geometric mean model and YieldSpace model: users can sell 10,900 PTs
- For Notional model: users can sell 102,936 PTs

For the same liquidity depth and interest rate change, Notional's AMM can accommodate a trade size of 9.44 times larger. As such, at t=1, the Notional model is 9.44 times more capital efficient than the other two models in this scenario.

Table 1 shows the numbers for when t=0.5 and t=0.25 as well, which shows that:

- The Notional AMM is consistently more capital efficient than the other models. It's around 9-35 times more capital efficient than geometric mean, and 8-9.4 times more capital efficient than YieldSpace's.
- The geometric mean model performs significantly worse than the other two models as it approaches expiry.

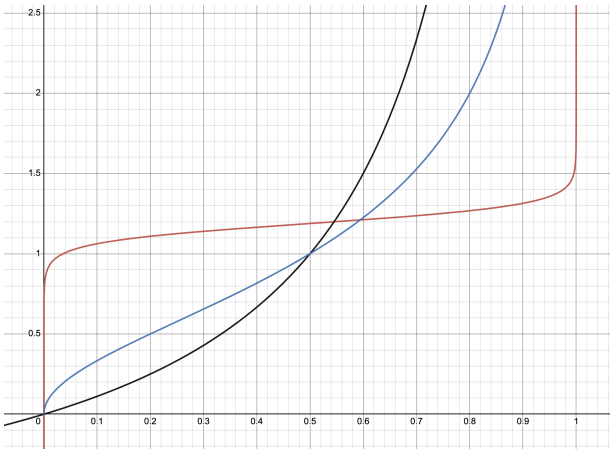

Figure 3: Scenario 1, t=0.5

Table 1: Scenario 1

t	1 (2 years left)	0.5 (1 year left)	0.25 (6 months left)
Estimated anchor rate	1.1881	1.09	1.034
Market interest rate	1.09	1.11	1.07
Desired interest rate	1.11	1.13	1.09
Possible trade size(Geometric mean)	10900 PTs	4977 PTs	2400 PTs
Possible trade size(Yield Space)	10900 PTs	9920 PTs	9567 PTs
Possible trade size(Notional)	102936 PTs	87671 PTs	83300 PTs

Figure (3) shows how the 3 models look like at t=0.5.

Scenario 2: Fixed yield trading for an asset X with very high yield:

- The expiry is in 3 months. $t_{start} = 1$ and $t_{oneYear} = 4$
- Average interest rate $rate_{expected} = 100$ (10000%)
- Max implied interest rate $rate_{max} = 200$ (20000%)
- Assume the pool has 1 million X worth of PT and X

For the Notional AMM, following the heuristics defined, we get initial rate anchor = 3.162, $rateScalar(t_{start}) = 1.0161$

Table 2: Scenario 2

t	1 (3 months left)	0.667 (2 months left)	0.333 (1 month left)
Estimated anchor rate	3.162	2.189	1.455
Market interest rate	100 (10000%)	110 (11000%)	90 (9000%)
Desired interest rate	110 (11000%)	120 (12000%)	100 (10000 %)
Possible trade size(Geometric mean)	18950 PTs	7964 PTs	3201 PTs
Possible trade size(Yield Space)	18950 PTs	11484 PTs	8336 PTs
Possible trade size(Notional)	29420 PTs	15121 PTs	9290 PTs

We can observe that: the Notional AMM model is still consistently more capital efficient. However, the difference is not as significant as in Scenario 1, around 1.5-2.9 more capital efficient than the geometric mean model and 1.1-1.5 more capital efficient than the YieldSpace model.

Scenario 3: Mimicking fixed yield trading for stETH (Liquid staked ETH in Lido):

- The expiry is in 1 year. $t_{start} = 1$ and $t_{oneYear} = 1$
- Average interest rate $rate_{expected} = 1.04$ (4%)
- Max implied interest rate $rate_{max} = 1.07$ (7%)
- Assume the pool has 1000 stETH worth of PT and stETH

For the Notional AMM, following the heuristics defined, we get initial rate anchor = 1.04, $rateScalar(t_{start}) = 54.93$

Table 3: Scenario 3

t	1 (1 year left)	0.5 (6 months left)	0.25 (3 months left)
Estimated anchor rate	1.04	1.0198	1.455
Market interest rate	1.04 (4%)	1.05 (5%)	1.03 (3%)
Desired interest rate	1.05 (5%)	1.06 (6%)	1.04 (4 %)
Possible trade size(Geometric mean)	2.494 PTs	1.22 PTs	0.609 PTs
Possible trade size(Yield Space)	2.494 PTs	2.43 PTs	2.43 PTs
Possible trade size(Notional)	136.6 PTs	116.48 PTs	130.5 PTs

We can observe that: the Notional AMM model is still consistently more capital efficient. However, the difference is significantly bigger than in Scenario 1. It is around 55-214 times more capital efficient than the geometric mean model and 48-55 more capital efficient than the YieldSpace model.

1.3.4 Conclusions

We can draw a few conclusions from this study:

- The Notional AMM model is consistently the most capital efficient. Intuitively, the two degrees of freedom in the Notional AMM model (in the form of the rate scalar and rate anchor) make each pool more specialised in trading principal token for a particular yield bearing asset.
- The geometric mean model is consistently the least capital efficient. This model becomes much less efficient compared to the other two when nearing expiry. Intuitively, this is because the model does not take into account the fact that PT price will converge to 1 at expiry.
- The YieldSpace model behaves exactly the same as the geometric mean model at the start, and becomes more capital efficient as it approaches expiry.
- The gap in capital efficiency between the Notional AMM model and the other models is the most pronounced for yield bearing assets with a tighter range of possible interest rates. Intuitively, the curve can be customised to be concentrated on a tighter range of interest rates, which makes the trading much more capital efficient.

From this study, we have decided to adopt the Notional AMM model for PendleV2's principal token AMM, with some slight changes. We will introduce how PendleV2 PT AMM works in the next sections

2 Pendle V2 PT AMM - Basic definitions

market is a pool of SY and PT, where users could trade one for the other, or provide liquidity to it

SY is a Standardized Yield token (as defined in a separate paper). At time t, the Market has $n_{sy}(t)$ SY tokens.

PT is a PT token of the same SY token (as defined in the Yield Tokenisation paper). At time t, the Market has $n_{pt}(t)$ PT tokens.

asset is the asset of the SY token.

totalAsset(t) is equivalent amount of asset that the SY tokens in the Market are worth, at time t. Short form: is $n_{asset}(t)$

$$n_{asset}(t) = n_{sy}(t) * syExchangeRate(t)$$

expiry is the expiry of the PT token. Short form: t_{expiry}

timeToExpiry(t) is the time left until the expiry

$$timeToExpiry(t) = t_{expiry} - t$$

yearsToExpiry(t) is the time left until the expiry, counted in years

$$yearsToExpiry(t) = timeToExpiry(t)/(oneYearDuration)$$

proportion(t) is a measure of the proportion of PT in the market at time t

$$p(t) = \frac{n_{pt}(t)}{n_{pt}(t) + n_{asset}(t)}$$

scalarRoot is a fixed parameter for each market, to adjust the capital efficiency of the market. It is the value for rateScalar

initialAnchor is the initial rate anchor to anchor the market's formula to be more capital efficient around a certain interest rate.

feeRateRoot is a fixed parameter for each market, which is the fees rate in terms of interest rate

rateScalar(t) is a parameter used in calculating the exchangeRate at time t, to adjust the capital efficiency.

$$rateScalar(t) = \frac{scalarRoot}{yearsToExpiry(t)}$$

rateAnchor(t) is a parameter used in calculating the exchangeRate at time t, to adjust the interest rate around which the trading will be the most capital efficient.

exchangeRate is the spot exchange rate of asset in PT, without any fees. At time t, it is $exchangeRate(t)$. Since 1 PT can be redeemed for 1 asset worth at the expiry, the value of 1 PT should naturally be less than or equal to 1 asset: $exchangeRate(t) \geq 1$

$$exchangeRate(t) = \frac{ln(\frac{p(t)}{1-p(t)})}{rateScaler(t)} + rateAnchor(t) \tag{14}$$

implied rate The **implied rate** for a certain exchangeRate $exchangeRate_0$ at time t, is the return rate per annum (for an interest rate of 5 % per annum, the return rate is 1.05) being implied by valuing PT at $\frac{1}{exchangeRate_0}$ asset. With a capital of 1 asset = $exchangeRate_0$ PT, we will receive $exchangeRate_0$ asset, after a duration of $timeToExpiry(t)$. As such:

$$impliedRateForExchangeRate(exchangeRate_0, t) = exchangeRate_0^{\frac{oneYearDuration}{timeToExpiry(t)}} \tag{15}$$

$$= exchangeRate_0^{\frac{1}{yearsToExpiry(t)}} \tag{16}$$

Conversely, at time t, an implied rate of $impliedRate_0$ is equivalent to a exchangeRate of:

$$exchangeRateForImpliedRate(impliedRate_0, t) = impliedRate_0^{yearsToExpiry(t)} \tag{17}$$

liquidity token represents ownership in the market. At time t, there is a total of $L(t)$ liquidity tokens.

liquidity providers are users who contribute liquidity to the market. At time t, a liquidity provider u has $l_u(t)$ liquidity tokens. It follows that:

$$\sum_u l_u(t) = L(t) \tag{18}$$

3 Pendle V2 PT AMM - State changes

We denote $t*$ as the time moment just before t when a certain event happens.

3.1 Initialisation

When a market is initialised at time t_0:

- $scalarRoot$, $initialAnchor$ and $feeRateRoot$ are set
- The following initial values are defined:

$$L(t_0) = 0$$
$$n_{sy}(t_0) = 0$$
$$n_{pt}(t_0) = 0$$
$$n_{asset}(t_0) = 0$$

3.2 Bootstrap liquidity

At time t, when $L(t*) = 0$, a user u can add d_{sy} **SY tokens** and d_{pt} PT into the market to bootstrap it. A portion of L_{locked} of **liquidity token** is locked forever in an pseudo-user u_x, such that $L(t)$ will never be 0 again and the market can only be bootstrapped once.

$$
\begin{aligned}
d_{asset} &= d_{sy} \times syExchangeRate(t) \\
L(t) &= d_{asset} \\
n_{sy}(t) &= d_{sy} \\
n_{pt}(t) &= d_{pt} \\
n_{asset}(t) &= d_{asset} \\
l_u(t) &= d_{asset} - L_{locked} \\
l_{u_x}(t) &= L_{locked}
\end{aligned}
$$

3.3 Add or remove liquidity

At time t, a user u adds d_l liquidity tokens worth into the pool (or removes from the pool if $d_l < 0$), where $d_l + l_u(t*) \geq 0$.
The user will need to add d_{sy} SY and d_{pt} PT into the pool (or can remove $-d_{sy}$ SY and $-d_{pt}$ PT from the pool if $d_l < 0$)

$$
\begin{aligned}
d_{sy} &= n_{sy}(t*) \times \frac{d_l}{L(t*)} \\
d_{pt} &= n_{pt}(t*) \times \frac{d_l}{L(t*)} \\
l_u(t) &= l_u(t*) + d_l \\
L(t) &= L(t*) + d_l \\
n_{sy}(t) &= n_{sy}(t*) + d_{sy} \\
n_{pt}(t) &= n_{pt}(t*) + d_{pt} \\
n_{asset}(t) &= n_{sy}(t) \times syExchangeRate(t)
\end{aligned}
$$

3.4 Trade

At time t, a user u buys d_{pt} PT from the market (or sells $-d_{pt}$ into the market if $d_{pt} < 0$).
First, we need to define a few terms:

lastImpliedRate The implied rate for the last trade that occurred at t_{last}.

trade proportion short form: p_{trade}

$$
p_{trade} = \frac{n_{pt}(t*) - d_{pt}}{n_{pt}(t*) + n_{asset}(t*)}
$$

trade exchangeRate with no fee short form: $exchangeRate_{tradeNoFee}$

$$
exchangeRate_{tradeNoFee} = \frac{ln(\frac{p_{trade}(t)}{1-p_{trade}(t)})}{rateScalar(t)} + rateAnchor(t)
$$

3.4.1 Interest rate continuity

To achieve interest rate continuity, we need to adjust the rateAnchor(t) such that the pre-trade implied rate, at $t*$ is the same as the $lastImpliedRate$

This means that:

$$\begin{aligned} impliedRateForexchangeRate(exchangeRate(t*), t) &= lastImpliedRate \\ \Rightarrow exchangeRate(t*) &= lastImpliedRate^{yearsToExpiry(t)} \end{aligned}$$

From (1), we get:

$$exchangeRate(t*) = \frac{ln(\frac{p(t*)}{1-p(t*)})}{rateScaler(t)} + rateAnchor(t) \tag{19}$$

We can hence deduce rateAnchor(t):

$$exchangeRate(t*) = \frac{ln(\frac{p(t*)}{1-p(t*)})}{rateScaler(t)} + rateAnchor(t)$$

$$rateAnchor(t) = lastImpliedRate^{yearsToExpiry(t)} - \frac{ln(\frac{p(t*)}{1-p(t*)})}{rateScaler(t)} \tag{20}$$

3.4.2 Dynamic fees charged on interest rate

Let's define $exchangeRate_{trade}(t)$ to be the actual exchangeRate for the trade, such that we are charging a fixed $feeRateRoot$ (for example, 1.01 for a 1 % fees) in terms of interest rate slippage. This means that:

$$\begin{aligned} &impliedRateForExchangeRate(exchangeRate_{trade}, t) \\ &\quad = impliedRateForExchangeRate(exchangeRate_{tradeNoFee}, t)(\times or \div) feeRateRoot \end{aligned}$$

($\times$ if the trade is from PT to asset)

Applying (15):

$$exchangeRate_{trade}^{\frac{1}{yearsToExpiry(t)}} = exchangeRate_{tradeNoFee}^{\frac{1}{yearsToExpiry(t)}} (\times or \div) feeRateRoot$$

$$exchangeRate_{trade} = exchangeRate_{tradeNoFee} (\times or \div) feeRateRoot^{yearsToExpiry(t)} \tag{21}$$

4 Pendle V2 PT AMM - Implementation details

4.1 Changing of variables for easier implementation

Since natural logarithms and exponents are easier to compute, we could introduce a few variables:

lnFeeRateRoot is $ln(feeRateRoot)$

lnLastImpliedRate is $ln(lastImpliedRate)$

(7) and (8) will become the following:

$$rateAnchor(t) = e^{lnlastImpliedRate \times yearsToExpiry(t)} - \frac{ln(\frac{p(t*)}{1-p(t*)})}{rateScaler(t)} \tag{22}$$

$$exchangeRate_{trade} = exchangeRate_{tradeNoFee} (\times or \div) e^{lnfeeRateRoot \times yearsToExpiry(t)} \tag{23}$$

4.2 Swapping exact amounts of assets

The AMM formula is defined in terms of the amount of PT being swapped, making it trivial to swap an exact amount of PT in or out of the pool. We can write the amount of asset going in d_{asset} as a function of amount of PT going out d_{pt}

$$
\begin{aligned}
d_{asset} &= f(d_{pt}) \\
&= \frac{d_{pt}}{exchangeRate_{trade}} \\
&= \frac{d_{pt}}{exchangeRate_{tradeNoFee}(\times or \div) feeRateRoot^{yearsToExpiry(t)}} \\
&= \frac{d_{pt}}{\frac{ln(\frac{p_{trade}(t)}{1-p_{trade}(t)})}{rateScalar(t)} + rateAnchor(t)} (\div or \times) feeRateRoot^{yearsToExpiry(t)} \\
&= \frac{d_{pt}}{\frac{ln(\frac{n_{pt}-d_{pt}}{n_{asset}+d_{pt}})}{rateScalar(t)} + rateAnchor(t)} (\div or \times) feeRateRoot^{yearsToExpiry(t)}
\end{aligned}
$$

For swapping an exact amount of asset in or out of the market, we need to use approximation algorithms because reversing $f(d_{pt})$ is non trivial.

4.2.1 Swapping asset for PT

Looking at the graph of $f(d_{pt})$ in Figure 4, we can see that for $d_{pt} > 0$, or when we are swapping asset to PT, f is a monotonous function. This means that we can simply binary search for d_{pt} given an exact d_{asset}.

One thing to note is that $exchangeRate_{trade}$ must be greater or equal to 1. As such,

$$
\begin{aligned}
exchangeRate_{tradeNoFee} \div feeRate &\geq 1 \\
\frac{ln(\frac{p_{trade}(t)}{1-p_{trade}(t)})}{rateScalar(t)} + rateAnchor(t) &\geq feeRate \\
ln(\frac{p_{trade}(t)}{1-p_{trade}(t)}) &\geq (feeRate - rateAnchor(t)) \times rateScalar(t) \\
\frac{p_{trade}(t)}{1-p_{trade}(t)} &\geq e^{(feeRate - rateAnchor(t)) \times rateScalar(t)} \\
\frac{1-p_{trade}(t)}{p_{trade}(t)} &\leq e^{(rateAnchor(t) - feeRate) \times rateScalar(t)} \\
\frac{1}{p_{trade}(t)} &\leq e^{(rateAnchor(t) - feeRate) \times rateScalar(t)} + 1 \\
\frac{n_{pt} - d_{pt}}{n_{pt} + n_{asset}} &\geq \frac{1}{e^{(rateAnchor(t) - feeRate) \times rateScalar(t)} + 1} \\
n_{pt} - d_{pt} &\geq \frac{n_{pt} + n_{asset}}{e^{(rateAnchor(t) - feeRate) \times rateScalar(t)} + 1} \\
d_{pt} &\leq n_{pt} - \frac{n_{pt} + n_{asset}}{e^{(rateAnchor(t) - feeRate) \times rateScalar(t)} + 1}
\end{aligned}
$$

二分查找法的基本思想

对于一个已知的单调递增函数 $y = f(x)$，如果我们知道 y 的值，想要求解对应的 x，二分查找法是一种非常有效的逼近方法。

- **单调递增**意味着当 x 增大时，$f(x)$ 也增大。
- **目标**是找到 x 使得 $f(x) = y$。

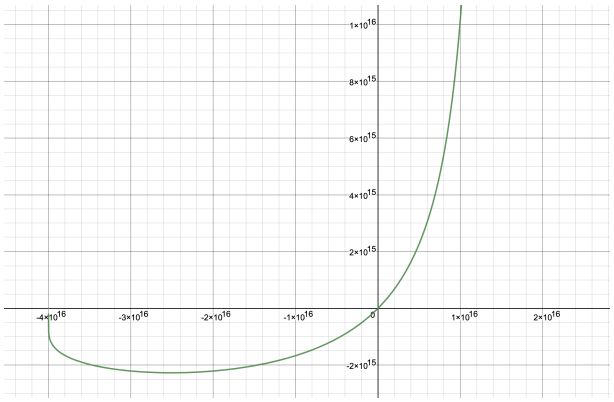

Figure 4: $f(d_{pt})$

With that, we have a bound on the maximum PT that could be bought, which is

$$n_{pt} - \frac{n_{pt} + n_{asset}}{e^{(rateAnchor(t) - feeRate) \times rateScalar(t)} + 1}$$

4.2.2 Swapping PT for asset

Looking at the graph of $f(d_{pt})$ in Figure 4, we can see that for $d_{pt} < 0$, or when we are swapping PT to asset, there is point of diminishing returns where swapping more PT will not get more assets.

This means that while binary searching for d_{pt} given an exact d_{asset}, we need to know whether we are on the negative or positive slope.

The sign of the slope for $f(d_{pt})$ is the same as the sign of the slope for $g(d_{pt}) = \frac{d_{pt}}{\frac{ln(\frac{n_{pt}-d_{pt}}{n_{asset}+d_{pt}})}{rateScalar(t)} + rateAnchor(t)}$, which is

$$slopeFactor(d_{pt}) = g'(d_{pt}) = \left(\frac{d_{pt}\left(n_{pt} + n_{asset}\right)}{\left(n_{pt} - d_{pt}\right)\left(n_{asset} + d_{pt}\right)} + \ln\left(\frac{n_{pt} - d_{pt}}{n_{asset} + d_{pt}}\right) \right) \cdot \frac{1}{rateScalar(t)} + rateAnchor(t)$$

Another note is that p_{trade} must be less than 1. Therefore, $n_{pt} - d_{pt} < n_{pt} + n_{asset}$, which means $-d_{pt} < n_{asset}$. In other words, the amount of PT that can be swapped in must be less than n_{asset}

References

[1] Hayden Adams, Noah Zinsmeister, and Dan Robinson. *Uniswap V2 Core*. URL: `https://uniswap.org/whitepaper.pdf`.

[2] Apwine. *Apwine AMM*. URL: `https://docs.apwine.fi/protocol/amm`.

[3] Balancer. *Balancer Whitepaper*. URL: `https://balancer.fi/whitepaper.pdf`.

[4] Element Finance. *The Element Protocol Construction Paper*. URL: `https://paper.element.fi/`.

[5] Notional Finance. *Notional AMM*. URL: `https://docs.notional.finance/notional-v2/technical-topics/notional-amm`.

[6] Sense Finance. *Sense Protocol Litepaper*. URL: `https://docs.sense.finance/litepaper/`.

[7] Vu Nguyen and Long Vuong. *Standardized Yield*. URL: `https://pendle.finance/SY.pdf`.
[8] Allan Niemerg, Dan Robinson, and Lev Livnev. *YieldSpace: An Automated Liquidity Provider for Fixed Yield Tokens*. URL: `https://yield.is/YieldSpace.pdf`.