CARBON

a decentralized protocol for asymmetric liquidity and trading

Whitepaper

November 2022 | v1.2

Carbon

a decentralized protocol for asymmetric liquidity and trading

Whitepaper

v1.2

Mark Richardson

mark@bancor.network

Stefan Loesch

stefan@bancor.network

Carbon Protocol

22 Nov 2022 (last updated: 7 Jan 2023)

Abstract

Carbon is a next-generation AMM protocol that introduces a number of innovative design features that greatly increase the possible use cases for this technology class. The key new feature introduced is that of asymmetric liquidity, where any given bonding curve only trades in a single direction, effectively being an out-of-the-money limit order. Two curves can then be linked together into more advanced trading and market-making strategies. Combined with a sophisticated routing and arbitrage design, Carbon expands the design space for trading and market making on-chain. This whitepaper describes Carbon's key features in detail.

carbondefi.xyz/whitepaper

Contents

1 Introduction: Carbon and the evolution of AMMs

Automated Market Makers – short "AMMs" – are key components of Decentralized Finance ("DeFi") infrastructure. AMMs allow market participants to trade their assets whenever they want, without the need to find another willing counterparty in the market, and without having to resort to centralized infrastructure.

Bancor and all other early AMMs used the *constant-product* technology. This technology is extremely elegant because it has no view on where the exchange rate of the two assets it trades should be – it works equally well everywhere. This advantage is also its greatest disadvantage, because in reality, constant-product AMMs work equally badly everywhere. The reason for this is that most of the time, any two assets in the market trade in well-known ranges. There may be periods of heightened volatility, but for reasonable time frames the maximum price changes of 10x, or possibly 100x are a far cry away from the complete scale invariance exhibited by constant-product AMMs which are designed to work equally well on ranges of the 1000x, 10000x or more. In other words – a lot of the collateral in constant-product AMMs is supporting trading at price points that will not be reached under any reasonable assumptions. This collateral is therefore employed extremely inefficiently and should be removed from the AMM.

1.1 Concentrated liquidity

The area where this problem was most pertinent was for like-kind assets, ie assets that are expected to trade at a constant (typically unity) price point. The best-known example for such like-kind assets are two stable coins that refer to the same underlying asset, for example USDC and USDT. Those are expected to trade in a very narrow range around unity, and any collateral that sits meaningfully away from unity will never be used. Also, if ever it is used this is probably a meltdown of one of the coins, and the AMM would have been better off halting the trading in this case to limit its losses. The desire to trade like-kind assets efficiently on AMMs led to the design of the so-called stable-swap invariant curves, where the large majority of the liquidity is deployed around unity. The purest example here is the constant-sum (or slightly more general, constant-price) curve where the entire liquidity is located at one specific price. The AMM then simply stops trading at either side of this price. In practice, stable-swap curves do accommodate a certain price range – a few percent – with the large majority of their liquidity, and they often also allocate a very small amount of liquidity to the wings so that they can continue trading (and continue to be useful for price discovery) without wasting much collateral.

Concentrated liquidity experienced a step-change with the introduction of Uniswap v3 in 2021: whilst stable-swap curves are designed to operate around a specific price point, the new model offers a multitude of concentrated trading curves, each associated with a specific trading interval, the entirety of which segments the price axis into non-overlapping and fully-exhaustive segments. A liquidity provider effectively invests in one, or multiple, of those stable swap curves. This however is a difficult choice, because only the segment corresponding to the current price point earns fees, so all collateral on the other segment is wasted. Also, liquidity cannot simply be moved: it must be withdrawn and re-staked, which can be a cumbersome and gas-intensive process.

1.2 Concentrated liquidity as trading facilitation

One interesting thing that started happening on Uniswap v3 was that people deliberately placed collateral far away from the money – not necessarily for liquidity provision, but rather with the intent of buying assets on a dip, or selling them after a rally. The traditional paradigm of AMMs was that liquidity providers were there to earn fees on their collateral, and that those fees are sufficient to not only cover for the so-called Impermanent Loss ("IL"), but also deliver attractive risk-adjusted returns over and beyond covering IL. We have shown previously that apparently this is not the case, and that the large majority of liquidity providers do not even break even once IL has been taken into account (see LHRW2021).

This traditional paradigm of AMMs is partially replaced by a trading paradigm, where "liquidity providers" no longer provide liquidity but instead use an AMM to submit limit orders to the market. However, using a traditional concentrated liquidity AMM for conditional trading runs into a number of issues. The first of those we already have alluded to above: when views change and positions are to be moved, that can be cumbersome and expensive, because a hefty cost is incurred every time a change is made rather than only when trading happens. More important however is the second issue: traditional AMMs provide symmetric liquidity, meaning one curve governs trading in both directions. So a liquidity provider may have placed an order to buy ETH on the downside, and may even have spent a lot on gas moving it around when expectations changed. Then the order is finally executed – but before the user, who may not be able to monitor it 24/7, has canceled it, markets may have moved back, and the position has traded back.

1.3 Carbon: asymmetric, parametrically adjustable, and concentrated liquidity

Carbon is a DEX designed with traders in mind, and its most important features are based on **asymmetric**, **parametrically adjustable**, and **concentrated liquidity**. We take those terms one by one. *Concentrated liquidity* has already been discussed above, and Carbon is not different from other AMMs, except that it can be used to create a wide range of bonding curves, from constant-product to constant-price, and everything in between. This is not fundamentally new, but it is covering the whole gamut of what is currently available in this space. *Parametrically adjustable liquidity* relates to what has been discussed above: rather than putting liquidity at a fixed place and having to redeem and re-stake it whenever change is desired, a Carbon "order" (our term for an individual user's liquidity position) is parameterized with an efficient set of parameters, and can be moved around quickly and cheaply simply by issuing transactions that update parameters.

The most important innovation in Carbon, however, is the asymmetric liquidity paradigm: contrary to all other AMMs on the market, Carbon's underlying curves trade in a single direction: one curve – or "order" – is either used for buying or for selling, but never for both. This is a very important point that we want to drive home: Carbon can essentially do everything other major AMMs can do, notably effectively-symmetric liquidity provision either across the whole curve, or within a range of any size, or even concentrated in a single point. But the main strengths of Carbon lie elsewhere, because it is fundamentally designed to support trading and market-making strategies composed of independent and irreversible buy and sell patterns.

Carbon offers two new primitives, and a number of others that are still under research. The first and most fundamental primitive is that of an **order** that consists of a single *parametrically adjustable, concentrated, and asymmetric* curve. Examples of orders are *"buy ETH for USDC in the range between 1500 (start) and 1000 (end)"* or *"sell ETH for USDC at 2000"*. The second primitive is that of a **strategy**, which consists of multiple, currently at most two, orders that can be linked together via their collateral. An example of a strategy would be the combination of the two orders above, leading to *"buy ETH for USDC between 1500-1000, and sell it at 2000; repeat until canceled"*.

1.4 Outline of this paper

In the remainder of this paper we will go through Carbon's feature set in more detail. We will start with Carbon's parametric liquidity, defining the set of hyperbolic curves we are using. We then discuss our main innovation, asymmetric liquidity, how we implement it in the case of a strategy composed of two linked curves, and some of its interesting properties, eg MEV resistance. Finally, we discuss how to adjust curves in a manner that does not require closing and recreating a liquidity position, thereby reducing the cost of active management.

Before we conclude we also briefly touch on an important issue that arises in the context of Carbon, which is how to optimally match trades to orders (what we refer to as *matching* and *routing*), and how to implement arbitrage strategies either within Carbon, or between Carbon and other systems. Those topics, however, are important enough to warrant their own publication and will be discussed in more detail in forthcoming publications.

2 Parametric concentrated liquidity

In this section, we discuss Carbon's first big innovation – **parametric concentrated liquidity** based on a set of recently and purposefully engineered invariant curves.

2.1 Pool invariant curve

Our generic pool invariant curve is

(desmos)

$$y = \frac{P_0 x_0 \left(x\left(\Gamma - 1\right) - x_0\left(\Gamma - 2\right)\right)}{\Gamma x - x_0\left(\Gamma - 1\right)} \tag{2.1}$$

Before we go further, we need to define the variables. As usual, x, y describe the state of the pool, with x to be interpreted as the amount of risk asset, and y as the numeraire. The variables P_0, x_0 and Γ are parameters whose meaning we will discuss further below.

The equation is of the form $y = f(x)$, and it is not immediately obvious that it is symmetric vis-a-vis exchanging the risk asset and the numeraire. We assert that it is, and we are here giving the reverse invariant function which reads as follows

$$x = \frac{y_0 \left(y\left(\Gamma - 1\right) - y_0\left(\Gamma - 2\right)\right)}{P_0\left(\Gamma y - y_0\left(\Gamma - 1\right)\right)} \tag{2.2}$$

Note that we used y_0 instead of x_0, reflecting our change of numeraire. We did not change P_0 however. Had we used the inverse price $\bar{P}_0 = 1/P_0$ instead, then the functional form would have exactly been the same. In other words – all the equations we show in this document hold when we replace $x, x_0, \Delta x$ with $y, y_0, \Delta y$, respectively, and we replace P_0 with $\bar{P}_0 = 1/P_0$.

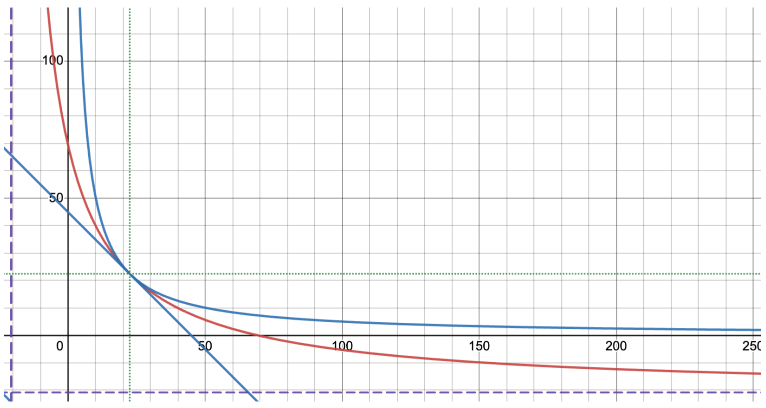


Figure 1: The new invariant curve

Also, because we will use it below, and because it is a very elegant form of describing the invariant curve, we want to introduce here another variant of that curve that is parameterized with the x and y intercepts x_{int}, y_{int} and the parameter $Q = (1 - \Gamma)^2$. Please see the appendix for a more thorough discussion and other relevant formulas.

$$Q = \frac{xy}{(x - x_{int})(y - y_{int})} \tag{2.3}$$

2.1.1 Geometry of the invariant curve

Whilst it is not immediately obvious when looking at the new invariant curve (the red curve in the figure "The new invariant curve"), its construction process has been very geometric. Fundamentally, it is still the same $x \cdot y = k$ hyperbola first introduced by Bancor and ultimately popularized by Uniswap. However, it has been translated in the x, y plane and has also been scaled. This yields 3 free parameters, 2 for the translation and 1 for the scale.

We have plotted the invariant curve in the chart "The new invariant curve", and we have also prepared an interactive version on Desmos that we invite the reader to play with. In the chart, the red curve is the curve for a specific value of Γ (about 0.5 in this case). The blue curve is the constant-product curve, and the straight blue line is the constant-price curve. The point where they all meet (also indicated by the green lines) is x_0, y_0, and the slope (aka price) at this point is P_0. The red curve is stretched around the fixed point x_0, y_0, keeping the slope fixed, and it no longer has the axis' as asymptotes, but the dashed purple lines instead. If we change the Γ value on Desmos, we can see how the curve starts from being identical to the constant-product curve at $\Gamma = 1$ to being identical to the constant-price curve at $\Gamma = 0$.

(desmos)

We have just seen that the new curve covers the constant-product and constant-price cases, depending on the parameter Γ, and we want to discuss this in more detail here.

2.1.2 Constant-price ($\Gamma = 0$)

First, we set $\Gamma = 0$. In this case, the pool invariant curve simplifies dramatically to

$$y = P_0\left(-x + 2x_0\right) \tag{2.4}$$

We see that this invariant function is of the *constant-price* form $x + Py = k$. This also shows that the parameter P_0 can indeed be interpreted as a price, at least for $\Gamma = 0$ where it is the *constant* price of the AMM. We also note that the expression $x + Py$ is the total value of the liquidity pool, expressed in units of the risk asset. This allows us to interpret the term x_0 as the invariant size parameter of the pool (expressed in terms of the numeraire), because for $y = 0$ we find $x = 2x_0$. In other words – x_0 is half the invariant pool size, expressed in terms of the numeraire.

2.1.3 Constant-product ($\Gamma = 1$)

We now set $\Gamma = 1$. Again, the equation simplifies dramatically, and we obtain

$$y = \frac{P_0 x_0^2}{x} \tag{2.5}$$

This is the *constant-product* invariant $xy = k$ where $k = P_0 x_0^2$. If we remember the parameter y_0 we previously introduced, then we can rewrite this as $k = x_0\, y_0$. So, for $\Gamma = 1$, the parameters x_0, y_0 are the respective pool holdings of the risk asset and the numeraire, at an arbitrarily chosen reference point, and P_0 is the price at this point. So, again, P_0 is a price-related parameter, and x_0, y_0 and $\sqrt{x_0 y_0}$ are all possible pool-size-related parameters.

2.2 Swap and price equations

From the pool invariant curve, we obtain the swap equation, ie the equation that describes what happens if we want to swap a quantity Δx of the risk asset into a quantity Δy of the numeraire, or vice versa. In other words, Δy is obtained as the difference of the right-hand side of the invariant curve when evaluated at x and $x + \Delta x$, respectively.

What we get in this case is the following equation

$$-\Delta y = \frac{P_0 \Delta x x_0^2}{\left(\Gamma\left(x - x_0\right) + x_0\right)\left(\Gamma\left(\Delta x + x - x_0\right) + x_0\right)} \tag{2.6}$$

or, equivalently, after slight refactoring

$$-\Delta y = \frac{P_0 \Delta x x_0^2}{\left(\Gamma x - x_0\left(\Gamma - 1\right)\right)\left(\Gamma\left(\Delta x + x\right) - x_0\left(\Gamma - 1\right)\right)} \tag{2.7}$$

Note the minus sign on the left-hand side, which indicates that when the AMM receives the numeraire, it returns the risk asset and vice versa. Therefore Δx and Δy always have the opposite sign.

The above equation can look somewhat daunting, but it is important to understand that, seen as a function $f : \Delta x \to \Delta y / \Delta x$, the structure is simple. Its general form is

$$-\frac{\Delta y}{\Delta x} = \frac{a}{1 + b\Delta x} \tag{2.8}$$

which importantly only depends on two parameters, a and b. This suggests that our curve space is somewhat over-parameterized, and that there are many different ways of achieving equivalent swap curves. This of course is not new: invariant curves generally contain too much information. For example, $\sqrt{k} = \sqrt{xy}$ is equivalent to $k = xy$ and so is almost any $f(k) = f(xy)$ for that matter.

Also, for the eagle-eyed, we need to point out that the (a, b) parametrization above breaks down for $\Gamma = 1$ in which case we could use $a\Delta x/b + \Delta x$. This in turn breaks down for $\Gamma = 0$. This is the well-known problem of parameterizing a mathematical manifold – we may need more than one *map*. We could introduce 3 variables, but this would obfuscate the fact that the function space is of dimension 2.

We now look again at what happens for the special values of Γ, and we first look at $\Gamma = 0$:

$$-\Delta y = P_0 \Delta x \tag{2.9}$$

This is a constant-price swap equation where the exchange between risk asset and numeraire always happens at a price P_0. When we look at $\Gamma = 1$ we find the well-known swap equation for the constant-product case:

$$-\Delta y = \frac{P_0 \Delta x x_0^2}{x\left(\Delta x + x\right)} \tag{2.10}$$

The swap equation relates macroscopic transactions $\Delta x, \Delta y$ to their effective price $P_{eff} = -\Delta y/\Delta x$. From this, we can move to the infinitesimal transaction dx, dy by going to the limit $\Delta x \to 0$. The effective price then becomes the **marginal price** $P_{marg} = -dy/dx$. This marginal price can also be obtained from the invariant curve above via derivation, so the term $-dy/dx$ can be rightfully interpreted as a derivative.

(desmos)

$$P_{marg} = \frac{P_0 x_0^2}{\left(\Gamma\left(x - x_0\right) + x_0\right)^2} \tag{2.11}$$

2.3 Leverage in Carbon

2.3.1 Methods of generating leverage in AMMs

The main purpose of Carbon's concentrated parametric liquidity is just that – concentration, aka leverage. Before we go into details on how Carbon creates concentrated liquidity, we remind ourselves how AMMs usually do that. In fact, there are two fundamental models:

1. Along the lines of Bancor v2 (but not v2.1) and Uniswap v3, by running the AMM on a *virtual* token supply, thereby creating bona fide leverage
2. Along the lines of Curve and others, by modifying the actual invariant curve

Carbon effectively uses a mix of both, so we will discuss the two methods in turn. Starting with the first option, what is a "virtual token supply"? We assume a standard constant-product AMM with 100m in liquidity. We then remove some tokens, but without telling the AMM about it, so it still trades as if it had 100m liquidity – up to the point it runs out, at which point it stops trading in the direction in which it can no longer trade.

To put it more formally, under the *virtual tokens paradigm*, the parameters x, y that are used in the invariant curve do not correspond to the actual token holdings x_{act}, y_{act}. Typically, we have $x_{act} < x, y_{act} < y$ because otherwise there are a number of "dead" tokens the AMM can never reach – in other words, the AMM is *delevered.* We remind ourselves that x is the risk asset, so $x_{act} < x$ means the AMM runs into a hard stop when it is trying to sell the risk asset. Therefore, the number x_{act} determines the *upper* boundary of the AMM's trading interval. Likewise, $y_{act} < y$ corresponds to the numeraire and therefore to purchases of the risk asset, so y_{act} determines the *lower* boundary of the interval. Upper and lower are to be understood in the numeraire where the asset being traded is considered the risk asset of the pair.

Changing the curve is, well, changing the curve. In this context we remind ourselves that

- the slope of the curve (tangent) corresponds to the marginal price at this particular point of the curve
- a connection between two points on the curve (secant) corresponds to a trade, with Δx being the amount of risk asset traded, Δy the amount of numeraire traded, and $\Delta y/\Delta x$ being the effective price
- the amount of liquidity in a certain segment of a curve is determined by its curvature, with flatter curves corresponding to more liquidity held in that price range.

The latter points merit more explanation because they may not be entirely intuitive. We consider an interval on the x axis, and the corresponding curve segment. The length of the interval on the x-axis is the amount of risk asset "contained" in this interval. The slope of the curve at every point is the marginal price, so if the curve is a straight line the *entire* liquidity is located at the same price point (the so-called "constant-price" curve). The more the slope on the left-hand side of the interval is bigger than the slope on the right-hand side (it can never be smaller as the curve is convex), the wider the price range covered by the same amount of risk asset liquidity.

Carbon does something in between those two models: it changes the curve, but in its current incarnation the curve change is equivalent to a virtual token model. The difference is that, whilst the virtual token model puts (x, y) on a different point of the curve, Carbon keeps (x, y) tied to financially meaningful numbers, but moves the curve around and stretches it. This can be seen nicely in the aforementioned Desmos model.

Financially this does not make a difference – yet. We are working on extensions of the model that would generate entirely different curves. The main difference is technical, which is important on the highly resource-constrained Ethereum chain. Implementation of this curve is extremely gas efficient, and it can be combined with an efficient routing framework. The parametric form of the curve allows for highly efficient implementation, and it also unlocks new composability paradigms that are hard to achieve otherwise.

2.3.2 Carbon leverage

We now look at how Carbon leverage works in practice. First, we go back to the claim that despite the apparent complexity of the formula, it is financially equivalent to the virtual token model. For this we recall that the swap equation – which is the authoritative record of the financial reality of this system – had effectively a very simple form along the lines of $a/(\Delta x + b)$, ie the curves only form a space of dimension 2. One of those dimensions is frozen once we fix the price, so at any given price there is only one free parameter left for the curve shape, and the set of curves attainable is exactly the same as the one under virtual token model.

Again, the eagle-eyed will run into one problem – the virtual token model cannot create the constant-price curve. It can get arbitrarily close to it, but it can never attain it. This is the aforementioned well-known manifold problem – it turns out that the "map" spanned by the virtual token model cannot represent the constant-price model because this is the "point at infinity". Which is another reason why our curve is superior – by over-parameterizing the problem with a redundant parameter set, we manage to cover the entire space.

Before we discuss Carbon liquidity in more detail, we need to point out that we are not yet dealing with the asymmetric liquidity paradigm that Carbon is also implementing. In this section we assume a standard, bi-directional curve, and we will discuss the changes that stem from the unidirectional liquidity provision in the next section.

For this discussion, we are looking at the **Q-and-intersect parametrization** of the invariant curve that we recall as

$$Q = \frac{xy}{(x - x_{int})\,(y - y_{int})} \tag{2.12}$$

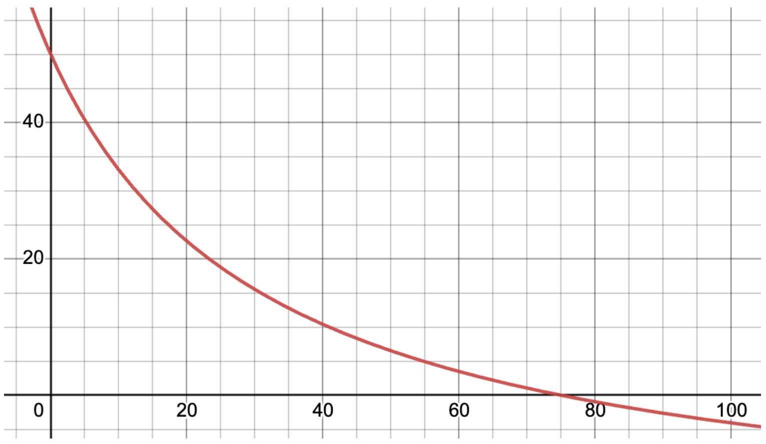


Figure 2: Carbon invariant curve (Q=0.3)

We have plotted this invariant curve for reference in the chart “Carbon invariant curve (Q=0.3)”, but we suggest looking at the interactive version on Desmos.

In the Carbon environment, the values of x, y correspond to actual token holdings. As those holdings cannot go negative, the AMM stops trading in one direction when it hits either of the axes. Therefore, the curve shown in the chart starts at 75 units of the risk asset (x-axis) and 0 numeraire units on one side, and ends at 50 numeraire units (y-axis) and 0 risk asset units on the other side. The average transaction price is $\frac{50}{75} \simeq 0.66$. It is somewhat hard to read from the chart here, but from the interactive version on Desmos we find that the slope where it cuts the y-axis $P_y \simeq 2.2$ and the one at x is $P_x = 0.2$, so this particular curve covers a wide price range. It turns out that curves that cover reasonable trading ranges look almost straight to the eye.

As for the original chart, $Q = 0.3$ is relatively close to the constant-product case $Q = 0$. As reference we have also drawn the same curve with $Q \simeq 0.8$ on the chart “Carbon invariant curve (Q=0.8)” which can also be seen on Desmos. This curve yields a price range of $0.5 - 0.8$ approximately, ie a range width of 1.5x.

We note en-passant that $Q = (1 - \Gamma)^2$, $P_0 = \sqrt{P_x P_y}$, and $P_y/P_x = 1/Q^2$, all of which are detailed in the appendix.

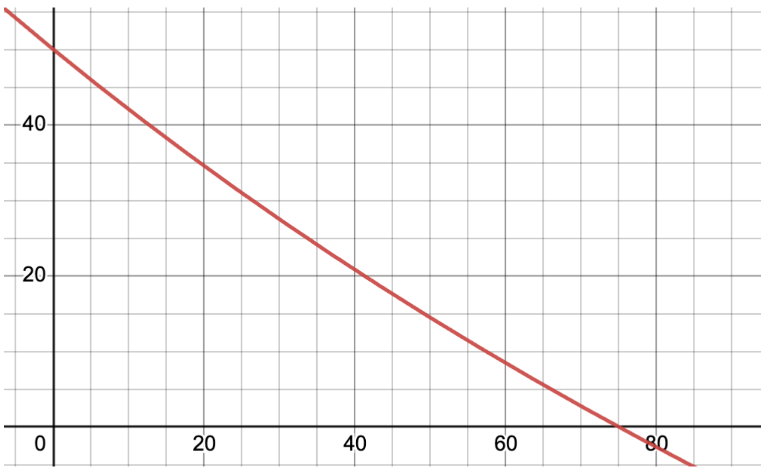


Figure 3: Carbon invariant curve (Q=0.8)

2.3.3 Leverage, pool prices, and the asymptotic invariant equation

There are a number of occasions where it is useful to quantify leverage, one key example being when we want to calculate the *effective price* of a pool. As will be discussed in more detail in the section on matching and routing, Carbon is unusual in that the matching algorithm, ie the algorithm that finds the best Carbon orders for a given trade, is non-trivial. In a traditional AMM, if we even think about an AMM as being separated into per-position pools, all those pools are set at the same price, by design. Now this price may or may not be the current market price, provided such price meaningfully exists in the first place, but at least all positions in the AMM agree what their current marginal price is.

For Carbon this is different. Even when we are only looking at the *active* orders, ie the orders where we have the strict inequalities $0 < y < y_{int}$ and the price is not stuck at the boundary, there is no guarantee that the marginal prices of all relevant pools are the same. They will be the same if our standard routing algorithm has been used, and if the last traded was routed through all active pools. However, there is no guarantee that this is always the case.

So we can easily be, and often will be, in a situation, where the marginal prices of all active pools diverge somewhat, and we would like to quantify something akin to the *average price*, as well as the *price divergence*. A naive analysis of this problem may suggest to simply use average and standard deviation, and in principle this is not a bad idea, except that some pools may be very small, and some very big, so the correct way to calculate the average is to use some pool-size weighted average, for each trade direction separately. This measure is also incomplete, however, and may often be misleading. At the very least it should be augmented by the best price available in each direction. Ultimately, the gold standard measure in this situation is the full order book that determines the price for every possible trade size.

Be that as it may, it is clear that we need a good measure of the pool size. As it turns out, traditional constant-product pools have an excellent measure for pool size, which is the pool constant k from the invariant equation $k = x * y$. In fact, it is well known that the best measure for pool size is $\sqrt{k}$, because it is the only measure that has the correct scaling properties, ie the only measure that scales linearly in the token amounts.

We also know that Carbon curves are constant-product curves, just moved and, importantly, scaled. So what we need is that scaling factor. Once we have it, we can use it in our weighted average above. To calculate the scaling factor, we introduce yet another form of the pool invariant equation, its *asymptotic form*

$$(x - x_{asym})\,(y - y_{asym}) = \kappa \tag{2.13}$$

It has three parameters, the two asymptotes x_{asym}, y_{asym} and the pool constant κ (refer again to the chart “The new invariant curve” and to Desmos for a graph showing those asymptotes). The asymptotes are given as

$$x_{asym} = x_0 \left(1 - \frac{1}{\Gamma}\right) \tag{2.14}$$

and

$$y_{asym} = P_0 x_0 \left(1 - \frac{1}{\Gamma}\right) \tag{2.15}$$

which are negative numbers. If we substitute those into the above equation, and we also substitute y with the right hand side of our initial invariant equation, then we find that κ is

$$\kappa = \frac{P_0 x_0^2}{\Gamma^2} \tag{2.16}$$

The term on the top is $P_0 x_0^2 = x_0 y_0 = k$, ie it is equal to the pool constant k, and therefore we get

$$\sqrt{\kappa} = \frac{\sqrt{k}}{\Gamma} \tag{2.17}$$

In other words, the scaling factor on a Carbon order is $1/\Gamma$. We remind ourselves that $\Gamma = 1 - \sqrt{Q}$ and that $Q = 1/\sqrt{w}$ where $w = P_y/P_x > 1$ is the percentage width of the price range, or rather the greater-than-unity ratio of its two end points. So we finally get the scaling equation we were looking for

desmos

$$\frac{\sqrt{\kappa}}{\sqrt{k}} = \frac{1}{1 - \sqrt[4]{\frac{P_x}{P_y}}} \tag{2.18}$$

We have plotted the equation in log scale on Desmos, and also in the chart "Log leverage versus range percentage width". On the x-axis this the percentage width of the range, eg $[100, 105]$ would be at 5. On the y-axis we have plotted log leverage. For example, at a 50 range width, the leverage is $10^1 \simeq 10$.

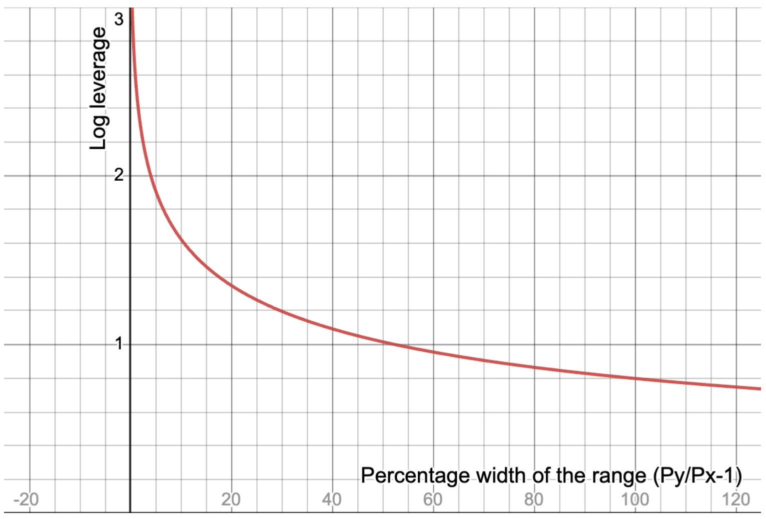


Figure 4: Log leverage versus range percentage width

3 Asymmetric liquidity

Carbon is designed as an AMM for traders and market makers, and in particular as a "limit order machine". There has been some discussion lately about using Uniswap v3 as such, by putting liquidity into a thin out-of-the-money bucket that will be traded when the prices go through the associated price interval. There are two issues associated with this approach

1. It is only possible to create one type of limit order, the "out-of-the-money" type, which is to buy the depreciating asset on the downside, or to sell the appreciating asset on the upside; it is not possible to buy an asset on its upside or to sell it on its downside.
2. When markets are jittering or range trading, the position may still be reversed when the markets move the other way; to avoid this, limit order positions need to be permanently monitored, and canceled immediately once they have been executed.

Carbon still has issue (1) – it is something intrinsic to AMMs, and it is not possible to address it in an AMM framework. So, "buy-low-sell-high" strategies can be implemented with AMMs, but stop-loss and buy-on-uptick ones cannot.

Carbon, however, fully solves issue (2). The way it does it is that every position has two curves, one for buying and one for selling. Those curves can in principle be the same, but they usually are not. So, one possible strategy is that if markets are at say 100, there can be a "buy" curve to accumulate the asset from 80 down to 50. And there can be a sell curve to sell from 150 to 200. Both curves *can* be present, but they do not *have* to be present – a Carbon position can simply terminate once executed, so there is only a buy order, or only a sell order.

3.1 Carbon asymmetric liquidity mechanics (simplified)

In this section we will go through a simplified version of the mechanics of asymmetric liquidity provision in Carbon. In the next section, we will then discuss the detailed mechanics.

Here, we will first discuss the mechanics with an example of a one-sided system (sell-curve only) that can be used as a sophisticated token distribution engine. We will then discuss an example of a generic "buy-low-sell-high" trading strategy.

3.1.1 Token distribution strategy

Our first example refers to the chart "Token distribution curves", but as usual we urge the reader to look at the version on Desmos as it allows for the reader to modulate the parameters.

(desmos)

- DC1 parameters (solid): $Q = 0.5, x_{int} = 500k, y_{int} = 500k$
- DC1 range (solid): $P_x = 0.5, P_y = 2.0, P_0 = 1$
- DC2 parameters (dashed): $Q = 0.87, x_{int2} = 500k, y_{int2} = 870k$
- DC2 range (dashed): $P_{x2} = 1.5, P_{y2} = 2.0, P_{02} = 1.7$

The chart has two curves, DC1 ("Distribution Curve 1") and DC2. We first look at DC1

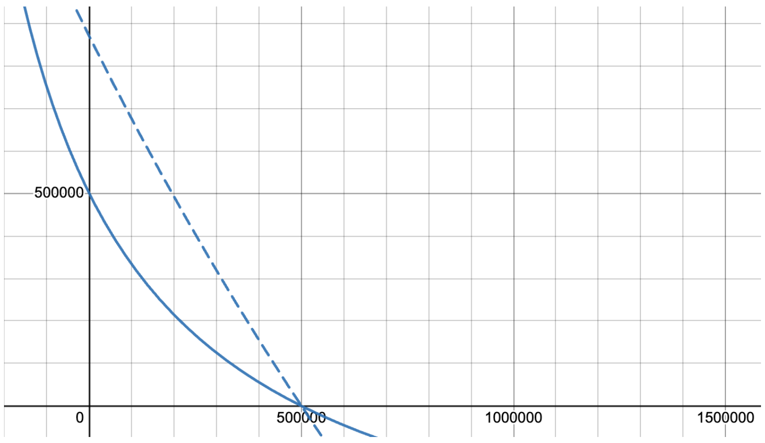


Figure 5: Token distribution curves (DC1: solid; DC2: dashed)

on its own. This curve contains 500k of risk asset tokens, and it sells them at an average price of 1, for 500k numeraire tokens. The price range is between $0.5 - 2$.

The curve DC1 on its own is a valid distribution strategy, but it does only allow for a limited number of choices – the price range and the amount. Therefore, we introduce a curve DC2 that is in addition to DC1. In this case, DC2 distributes another $500k$ tokens, but in a smaller range $1.5 - 2.0$, receiving up to $870k$ in numeraire.

In other words – DC1 and DC2 together implement a distribution strategy in the range $0.5 - 2.0$ that distributes a total of $1m$ tokens, against a total $1.37m$ of cash. The distribution process is back loaded, as the second half of the tokens is only distributed at 1.5 and above. Please refer to the workbooks `CarbonSim-LitepaperExamples.ipynb` and `CarbonSim-LitepaperExamplesShort.ipynb` in our Carbon Simulator for a simulation of this position using our open-sourced Python libraries.

3.1.2 Buy-low-sell-high trading strategy

Our second use case refers to the chart called "Buy-low-sell-high curves", that can also be found on Desmos.

(desmos)

For reference, the curve parameters are as follows

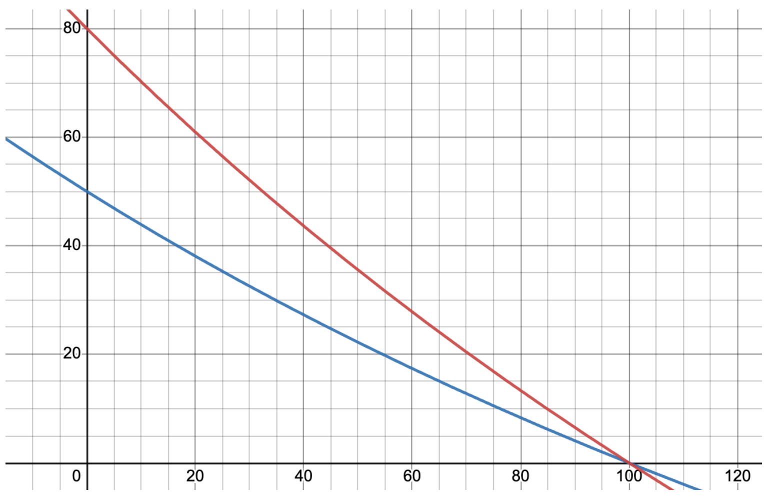


Figure 6: Buy-low-sell-high curves (buy: blue; sell: red) (blue)

- Buy (blue) parameters: $Q = 0.8, x_{int} = 100, y_{int} = 50$
- Lower range (buy; blue): $P_x = 0.4, P_y = 0.625, P_0 = 0.5$
- Sell (red) parameters: $Q = 0.8, x_{intr} = 100, y_{intr} = 80$
- Upper range (sell; red): $P_{xr} = 0.64, P_{yr} = 1.0, P_{0r} = 0.8$

Here, the blue curve is the *lower range*, ie the range where the risk asset x is *bought* (and the numeraire is sold), and the red curve is the *upper range* where the risk asset is *sold* (and the numeraire is bought).

We assume we start outside and above the buy range, and the price of the risk asset is falling. In this case, buying only starts once the range is entered, and buying stops whenever prices go back up. Importantly, there is no *selling* off the blue curve on the way up – it is simply paused. Buying resumes however whenever the prices go below its previous low water mark.

So, in the chart shown, the risk asset is *bought* starting at a price of $P_y = 0.625$ y-per-x, and ending at a price of $P_x = 0.4$ y-per-x. Once buying is finished – ie when the price drops to or below 0.4 – a total of $x_{int} = 100$ units of the risk asset will have been bought, against $y_{int} = 50$ units of cash, ie at an effective price of $P_0 = 0.5$ y-per-x. As a reminder,

$P_0 = \sqrt{P_x P_y}$ is the geometric average of the end prices of the range.

On the way up – the red curve – the situation is mutatis mutandis the same: the risk asset is *sold* starting at a price of $P_{xr} = 0.64$ y-per-x, and an end price of $P_{yr} = 1.0$ y-per-x. Once selling is finished – ie when the price rises to or above 1.0 – a total of $x_{intr} = 100$ units of the risk asset will have been sold, against $y_{intr} = 80$ units of cash, ie at an effective price of $P_0 = 0.80$ y-per-x.

So, if we started between the ranges with 50 units of the numeraire, and we went all the way down to below 0.4 and up again to 1 we end up with 80 units of cash, at a profit of 30.

This is a key use case of Carbon: the "liquidity provider" (we refer to them as "strategy providers") can use Carbon to run a **buy-low-sell-high** trading strategy. For this strategy to work, we need the two curves present, and we need $x_{int} = x_{intr}$ and $y_{int} < y_{intr}$. Also, the curvatures Q, Q_r must be chosen such that $P_y < P_{xr}$, ie so that the ranges do not intersect for $x, y > 0$.

Again, please refer to the workbooks `CarbonSim-LitepaperExamples.ipynb` for a longer discussion, and `CarbonSim-LitepaperExamplesShort.ipynb` for calculations-only, in our Carbon Simulator for a simulation of this position using our open-sourced Python libraries.

3.2 Actual Carbon asymmetric liquidity mechanics

After having described the *simplified* Carbon asymmetric liquidity dynamics above, in this section we discuss the *detailed* mechanics. Before we do that, however, we go through a brief example to show what the issue is with the simplified dynamics.

For this discussion, we are again referring to the chart "Buy-low-sell-high curves" that can also be found on Desmos. We assume we start somewhere between the ranges, eg at a price of 0.63. We are above the buy range, so as far as the blue curve is concerned, we need to be 100% in the numeraire. We are also below the sell range, so as far as the red curve is concerned, we need to be 100% in the risk asset. The only solution that satisfies both conditions is a zero position which is of course not particularly useful.

What we have just shown is that it is not possible to run both curves on a single shared state (x, y). Instead, we need two separate states (x^-, y^-) for the lower range, and (x^+, y^+) for the upper range. Note that the effective state space is of dimension 2, not 4, because of the two invariant curves that need to be satisfied that tie the x to their respective y.

Using Carbon conventions, the *active* asset on a curve (the one being *sold*) is always on the y-axis, and the *passive* asset is on the x-axis. There is a correspondence between *differences* but not between the absolute values of the cross-axis. This is best explained with an example. We examine the chart "Two linked curves", where the active token on the left-hand side is some asset RSK, and the active token on the right-hand side is USD (note that the risk asset / numeraire distinction does not make sense in Carbon). In this case we find that

1. The y-axis on the left-hand chart is denominated in RSK and corresponds to the actual number of tokens on the "RSK" (sell RSK and buy USD) curve
2. The y-axis on the right-hand chart is denominated in USD and corresponds to the actual number of tokens of the "USD" (sell USD and buy RSK) curve
3. The x-axis on the left-hand chart is denominated in USD; its absolute values do not matter, but differences are transmitted 1:1 to the right-hand side y-axis (2)
4. The x-axis on the right-hand chart is denominated in RSK; its absolute values do not matter, but differences are transmitted 1:1 to the left-hand side y-axis (1)

Note that the system only tracks the y-values of the respective charts, as the x-values can be recovered from the invariant curves.

The LHS ("sell RSK") chart contains up to 100 units of RSK, and it sells at a range around the average value of 225 USD per 100 RSK, ie 2.25 USD/RSK. The RSK ("sell

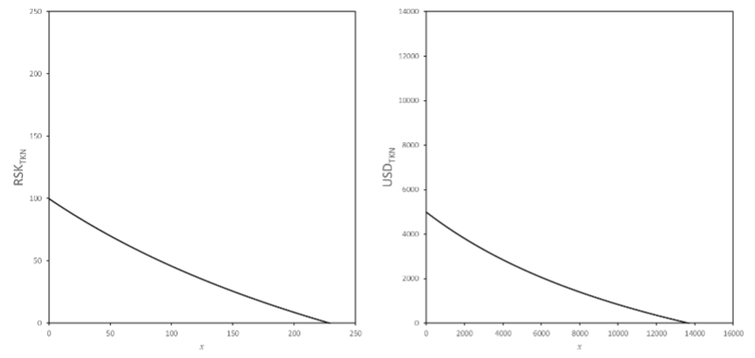


Figure 7: Two linked curves

USD") chart is much bigger – it contains up to 5,000 units of USD, and it buys RSK at an average price of 5,000 USD per 14,000 RSK, ie 0.36 USD/RSK. Note that those values are extreme for demonstration purposes – realistic values are usually somewhat closer together. However, the difference in volume is realistic: it may simply correspond to a "buy-low-sell-high" curve that has been seeded with USD and that did not get a chance of buying many RSK yet, so the RSK curve has not fully expanded at this point (see below for the meaning of "expanded").

The trade mechanics can be summarized by the following rules

1. Every token is sold on its own curve, where it appears on its y-axis
2. The y-value on a token's own curve corresponds to actual token holdings; selling stops at $y = 0$
3. Every transaction yields a $\Delta y < 0$ (sell the active asset) and a $\Delta x > 0$ (buy the passive asset)
4. The Δx obtained in a transaction is added to the y-axis on the other curve ($\bar{y}$) in a 1:1 ratio
5. In case the other curve does not have sufficient capacity, ie if $\bar{y} + \Delta x > \bar{y}_{int}$ in the other axis, then its capacity is increased by setting $\bar{y}_{int} := \bar{y} + \Delta x$
6. Curves only ever automatically expand, but they never shrink automatically; the

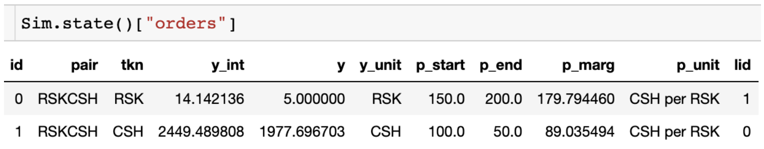
latter has to be done manually, if so desired

To summarize, each of the curves describes the sale – and the sale only – of its active asset (by convention the one on the y axis), and selling stops when the curve runs out of tokens at $y = 0$. The tokens obtained against the sale are moved onto the opposite curve, expanding it if need be. The system never shrinks a curve, as this could open arbitrage opportunities.

It may be worth going through a brief example with the following parameters: the lower range is $[50, 100]$, with the effective price being its geometric average 70.7. The upper range is $[150, 200]$ with effective price 173.2. If we start at a price between the ranges, say at 110, with $1,000$ units of cash, and we go to below 50, then the cash will have been converted into $1000/70.7 \simeq 14.1$ units of the risk asset. Those risk asset units move onto their own sell curve, and we assume it needed expansion (it may even have started at zero!) so now we have a curve that is fully loaded with 14.1 units of risk assets. If we move to 200, then those will be sold at an effective price of 173.2, yielding $14.1 * 173.2 \simeq 2,449$ units of cash that move to the other curve.

We now have $2,449$ units of cash which would correspond to $2,449/70.7 \simeq 34.6$ units of the risk asset if we'd go all the way down again. We assume, however, that we only dip into the lower order, to about 89, and we only get filled 5 units of the risk asset before prices go up again. The capacity of the risk curve is still 14.1 so at this stage the curve is not "full". This means that instead of starting to sell at 150 as a full curve would, the curve is top loaded, and selling will start closer to the end point, at 179.79.

We prepared the simulation workbook `CarbonSim-WhitepaperExample.ipynb` in our Carbon Simulator. The chart *"Carbon Simulator example"* shows the final state of the simulation, with `y` value of 5 in the first row indicating that the sell-RSK position contains 5 RSK, and the `p_marg` value of 179.79 indicating that this starts being sold at a price of 179.79. The second row shows the linked sell-RSK position, and the last column indicates that those two are indeed linked.



```
Sim.state()["orders"]
```

id	pair	tkn	y_int	y	y_unit	p_start	p_end	p_marg	p_unit	lid
0	RSKCSH	RSK	14.142136	5.000000	RSK	150.0	200.0	179.794460	CSH per RSK	1
1	RSKCSH	CSH	2449.489808	1977.696703	CSH	100.0	50.0	89.035494	CSH per RSK	0

Figure 8: Carbon Simulator example

3.3 Carbon deployment strategies and the role of fees

Taking a step back, the Carbon AMM can be fundamentally used for two different types of operations:

1. As a **fee-earning liquidity provider** to the market that is highly capital efficient due to the nimble deployment features discussed in a section below
2. As a **limit or range order machine** that allows traders to take a long-term view in range trading markets, or that assists market makers in making markets, and that are again greatly improved by the nimble deployment features discussed below

We have discussed the limit / range order functionality above, and we now briefly discuss fees here. There are two ways of thinking about fees: the conventional way is that they are a payment rendered for services provided that are independent of the transaction at hand. This commercial view, however, is not the full picture: in a trading context, fees are much better understood as a *bid-offer spread* that increases the price when a market maker is selling, and decreases it when it is buying. For example, assume markets are "at 100", and there are 1% fees. This means that a taker will pay 101 to buy (100 + 1 in fees), but will only receive 99 when selling (100 - 1 in fees). So fundamentally, fees describe a process where a liquidity provider buys and sells at different prices, just like any other market maker, and this is how they make money.

At the moment, Carbon does not allow for specifying an AMM as a mid-price plus fees. This has a practical reason: the routing algorithm is more efficient if we do not allow for bidirectional curves with fees, but we consider it as a future extension.

We note however that in principle it is possible to replicate the *mid-curve plus fees* paradigm in our model. In order to do this, the buy and sell curve must be very similar. More precisely, if the mid-curve is $P_a \ldots P_b$ then the buy curve is $P_a/(1+\alpha) \ldots P_b/(1+\alpha)$ and the sell curve is $P_a \cdot (1+\alpha) \ldots P_b \cdot (1+\alpha)$ for a fee level of 2α. Also, the two curves must be seeded with the correct amounts, so that the state of the two curves is correctly synched. Whether or not a Carbon UI allows for contributing such positions is a question left to the UI designers, but nothing prevents people from interacting directly with the contracts to create them.

3.4 Carbon and MEV

An important issue for AMMs is MEV, the "Miner Extractable Value", which are profits that parties who control the transaction flow – typically miners – can extract. An important MEV attack vector for AMMs is the sandwich attack, where a genuine transaction is sandwiched between transactions of the attacker. An AMM sandwich attack is very similar to frontrunning in traditional markets, except that a sandwich attack is guaranteed to either succeed, or to fail costlessly, other than gas. The way it works is as follows:

1. The attacker identifies a reasonably large trade order, eg for buying ETH against USDC; this order is a "market order", ie it fixes a USDC amount, and takes whatever amount of ETH it will get
2. The attacker inserts a large order buying ETH against USDC immediately before the attacked transaction
3. The attacker inserts an equal and opposite (apart from arbitrage gains) transaction to (2) immediately after the attacked transaction

What happens if the above is executed successfully is that the price at which the attacked transaction gets filled is artificially high, due to the price impact of the transaction (2). The transaction (3), which is now *selling* ETH, benefits from both the price impact introduced by the transaction (2) and that introduced by the attacked transaction.

Net/net, the attacker shifted the transaction to a higher price point off-market and can pocket that difference in price in a risk-free manner.

This particular attack vector is closed in Carbon, provided the asymmetric liquidity / two-curve pattern is used with curves that are non-overlapping, or at least far enough apart: whilst an attacker can still front-run a transaction as described under (2) above, the reverse transaction (3) will happen on the other curve, and therefore under vastly different conditions, making this particular attack vector no longer profitable.

3.5 Outlook

One final thing to mention here is that in this paper we restrict the choice of Γ to the interval $[0, 1]$, ie we allow for *constant-price*, *constant-product*, and curves that lie in-between those two. This is the reasonable choice for a market maker or trader, as for $\Gamma > 1$, collateral moves too far to the borders to be useful. We may investigate whether the increased convexity in a leveraged case – where the collateral that sits too close to zero and infinity is simply removed – is sensible for some use cases, but at the moment we do not believe it is. For $\Gamma < 0$, the curve becomes inverted, in the sense that when selling the risk assets, the first units are sold at a *higher* [sic] price than those that follow, and when buying them the first units are bought at a *lower* price, which reverses the order in an order book.

Whilst this does not make sense from a market-making perspective, there are some use cases where this can be interesting. Selling the first units more expensively is something like a "volume discount", and there may be an economic primitive for which this becomes interesting, eg for the token distribution use case discussed above.

4 Adjustable bonding curves

We have previously discussed how Carbon bonding curves are "parametric", in the sense that they are hyperbolic curves that are driven by a small number of parameters. When we first introduced those curves, we described them in terms of the parameters P_0 (a price-like parameter), x_0 (a size-like parameter, expressed in the risk asset), and Γ (a unit-less parameter describing the shape of the curve). We also have the curve state x (another size-like parameter expressed in the risk asset). We have also shown that, as expected, we can replace the risk asset denominated parameters x_0, x with the numeraire denominated ones y_0, y and the shape of the curve remains the same. In this discussion, we use the second form because it aligns more closely with the form that we actually use in the implementation, but of course the essence of the discussion does not depend on the parameterization of the curve. We remind ourselves of this second equation below

$$x = \frac{y_0\left(y\left(\Gamma-1\right)-y_0\left(\Gamma-2\right)\right)}{P_0\left(\Gamma y-y_0\left(\Gamma-1\right)\right)} \tag{4.1}$$

We also remind ourselves that in Carbon, liquidity is asymmetric. Because of this, the two tokens in a pool are *not* equivalent, as the one that is being *sold* defines the binding constraint: once the pool runs out of tokens to sell it stops, at least until it receives fresh tokens from a linked curve. In our convention, this “active” token is on the y-axis. As discussed in the section on AMM leverage, we use actual, not virtual, token balances, so the key constraint on the pool is $y > 0$.

If we ignore the constant-product boundary case (again, the manifold mapping problem), then we can describe our curves with three financial parameters

- the **start** of the range, where the AMM starts selling the active token, P_{start}; in our paramerization, this is the slope at the x-intercept, and is expressed with the active token as numeraire (“dy/dx”)
- the **end** of the range, where the AMM stops selling, P_{end}; this is the slope at the y-intercept, again expressed with the active token as numeraire
- the **capacity** of the curve, which determines how much collateral is needed to move from one end of the range to the other one; this is the y-intercept y_{int}

The *state* of the AMM is then described by y with $0 \leq y \leq y_{int}$, which corresponds to the *amount of tokens on-curve.* Taken together, these four parameters determine the entire trading curve of the AMM, and therefore in particular its *marginal price.* For $y = y_{int}$ the marginal price is $P_{marg} = P_{start}$ and for $y = 0$ it is $P_{marg} = P_{end}$. We always have $P_{start} \geq P_{end}$ which sounds counterintuitive until we understand that those P are expressed as dy/dx in the numeraire of the active asset, which inverts how we typically think about an asset that we are trading. If we consider y the risk asset and x the numeraire for a moment, then prices will be expressed as dx/dy, in which case the price at which selling starts is below the price at which selling ends, as it should be.

The importance of the marginal price is that if we are not careful here, we may create an unnecessary arbitrage opportunity for the market. We here assume that there are reasonably liquid markets elsewhere, so the *market price* is well established. Without loss of generality, we assume that that market price, in the reverse units, is currently at $\bar{P}_{mkt} = 100$ "dx/dy". The AMM is selling, therefore showing a price *below* 100 will yield an immediate arbitrage opportunity which we need to avoid.

In terms of starting position we need to consider two cases

1. The AMM is (deep) *out-of-the money*, ie the current market price is *far below* (in dx/dy) the starting price of the range, ie $\bar{P}_{mkt} \ll \bar{P}_{start}$.
2. The AMM is *in* (or *at*, or *close to*) the money, ie the current market price is *above* (or *equal to* or *close to*) that starting price of the range, ie $\bar{P}_{mkt} \geq \bar{P}_{start} - \epsilon$ where $\epsilon \geq 0$ is some suitably chosen proximity parameter.

In case (1), we have almost full freedom of how we adjust the range: the marginal price of the AMM is far away from the current market price, so we face little constraint in adjusting all parameters, including changing the state by adding or removing liquidity. In case (2) however, we need to be more careful: if $\epsilon = 0$, we are in the range, and we need to ensure that we keep the marginal price constant $\bar{P}_{marg;2} = \bar{P}_{marg}$ (if the AMM is to remain at-the-money) or at least non-decreasing $\bar{P}_{marg;2} \geq \bar{P}_{marg}$. If $\epsilon > 0$, then we have a little room of maneuver, meaning the marginal price can decrease up to ϵ, but other than that the same applies.

In practice, what this means is that in case (1), all 4 parameters – or rather, 3 parameters plus 1 state variable – can be changed independently, whilst in case (2), only 3 parameters can be changed independently if the AMM is meant to remain at the current marginal price. Therefore in this case, the allowable parameter space is in the 3-dimensional hyper surface embedded into the 4-dimensional parameter space. In case the condition is relaxed, and we allow for moving out-of-the money, then the allowable parameter space is a 4-dimensional half-space, ie it lies in a 4-dimensional space, on one side of the aforementioned 3-dimensional "constant marginal price" hyper surface.

We have just mentioned that the current state of the AMM, y, is part of this parameter set that needs to be carefully changed, yet we remind ourselves that strategies with linked curves do move collateral from one curve to the other which could be an issue. The way we currently avoid this is that the two linked curves (we do not currently allow more than two, and they must be on the same pair) must be non-overlapping. In this case, as long as one of the curves generates collateral to be placed on the other one, the target curve is forcibly out of the money, and therefore condition (1) applies. If we allow overlapping curves – or future strategies with more than two curves – however, this issue requires closer attention. We also want to point out that this is mostly a user interface issue: on the smart contract level (and possibly the API) we may well allow for adjustments that bring the contract into an arbitrageable condition. It is mostly important that users that make changes via a UI are protected here.

There are too many possible combinations of changes to consider covering them all, so the important thing to keep in mind that parameters always have to, at least, be changed in pairs if we are in case (2): we cannot just change one variable, we need to adjust (at least) one other variable as a response to that. We are going here through one example, which is looking at how to resize or move a range. For this we define $w = P_{start}/P_{end} > 1$ (we are back in the dy/dx numeraire) and we rewrite the marginal price equation as

$$P_{marg} = \frac{P_0 \left(-y\sqrt{\frac{1}{w}} + y + y_{int}\sqrt{\frac{1}{w}} \right)^2}{y_{int}^2 \sqrt{\frac{1}{w}}} \tag{4.2}$$

We remind ourselves that $P_0 = \sqrt{P_{start} \cdot P_{end}}$ corresponds to the location of the range. We can transform the above equation to

$$P_0 = \frac{P_{marg} y_{int}^2 \sqrt{\frac{1}{w}}}{\left(-y\sqrt{\frac{1}{w}} + y + y_{int}\sqrt{\frac{1}{w}} \right)^2} \tag{4.3}$$

which is of the functional form $P_0 = f(w)$. In other words, if we want to change the width w of a range (and we want to keep the other parameters untouched) then we have to change its location P_0 and vice versa.

5 Matching, routing, and arbitrage

In this section we briefly deal with matching, routing, and arbitrage. Before we go on, we need to define those terms. Matching and routing is about finding the best way to execute a given trade, and in our terminology, *matching* only looks within a given pool whilst routing also looks at trades that go through different pools and therefore involve three or more tokens. Arbitrage is about identifying imbalances in the system, both at the single pool level, and between pools, so we de facto have the same duality as in routing here even though we do not semantically acknowledge it.

5.1 Matching and routing

Matching is an exercise particular to Carbon because in other AMM designs it is not necessary. Or rather, it is so trivial that it is usually ignored. In order to understand this statement we need to take a step back and look at the theoretical foundations of AMMs. To make it somewhat more tangible, let us consider one of the most dramatic events in the history of AMMs, Sushiswap's vampire attack on Uniswap. A key fact is that SushiSwap cloned the Uniswap v2 (constant product) contracts and established identical pools to those of Uniswap, and then incentivized Uniswap liquidity to move (that's the vampire bit).

For the ease of presentation, we assume that Sushi cloned all pools, and that the conditions (most importantly, the fee levels), are exactly the same. We also assume negligible gas costs, which is a reasonable assumption at least for trades of $100,000 or above. In this case we have a routing problem (actually, a matching problem in our terminology) because if we want to trade, we need to decide how much to send through Uniswap and how much through Sushi. Fortunately, we have intermediaries like 1inch that determine the optimal path. We see that especially for bigger numbers (ie negligible gas cost), 1inch will show multiple routes for each trade.

1inch deals with a complex real-world problem, but for simplicity we'll return to our *two identical pools only* situation. In this case it is easy to show that the optimal routing is simply pro-rata to the liquidity in the respective pools. Now we can do a two-step reasoning with important insights. Step 1 is to understand that, from a trader's perspective and ignoring gas, Uniswap and Sushi are not really different entities. They just form part of a bigger "World AMM". This yields

> **Insight 1.** All AMMs available on a single network like Ethereum constitute the "World AMM" for this network (cross network introduces additional complexity because of non-atomicity).

Here we zoomed out to the big picture, so now we go the other way: if the World AMM in a world of only Sushi and Uniswap can be broken down into those two exchanges, surely any single AMM can be broken down further as well. This yields

> **Insight 2.** Any AMM can be considered as an aggregation of constituent AMMs that are formed by the individual user positions, with a routing (matching) algorithm to route trades to those constituent AMMs.

In other words, pre-Carbon, AMMs already have individual user positions. However, those positions are bucketed, rather than *personalized*, in order to ensure that routing is pro-rata and therefore trivial. This yields

> **Insight 3.** Pre-Carbon AMMs have been designed with the intent of making routing trivial, thereby impacting the user experience because users are forced into buckets. Carbon is designed in a manner that routing is still highly efficient, despite more personalized user positions.

Here we leave it at that because routing is (a) the center of ongoing research and (b) the subject of a forthcoming paper. Suffice to say that our routing mechanisms can already be explored in the Carbon Simulator that is available under an open-source license on our Github.

5.2 Arbitrage

Routing is the answer to the question *"I want to trade [amount] of [TKN1] to [TKN2]. What is the best way to do it?"*. Arbitrage is, in a narrower sense, the answer to the question *"I can trade [amount] of [TKN2] to [TKN1]. If I do the reverse trade somewhere else, will I make money?"*, so clearly those two are intimately related. Arbitrage however is more complex because it is the original problem "squared": In order to fully arbitrage a market we need to look at all possible optimal routes (side 1 of the square) and then look at all other possible routes (side 2 of the square) to see whether there is money to be made. Now we are not particularly interested in arbitraging the whole world – well, we are, but this is hard – so in our current research we focus on two questions:

1. Are there transactions within Carbon – ie transactions that connect two or more Carbon orders – that yield a positive return after fees and gas?
2. Are there transactions that allow me to arbitrage Carbon against the external market?

Again, those are matters of active research and are subject to a forthcoming paper, and we will also release a simple arbitrage bot in due course. If you are interested in discussing those topics with us before that, please do not hesitate to get in touch with any of the authors.

6 Conclusion

In this paper, we discussed Carbon, a next-generation AMM protocol that pushes the boundaries of what AMMs can do, and that opens a whole new world of applications, notably in trading and market making. Carbon's key design features are:

- **Asymmetric liquidity.** Asymmetric liquidity means that each trading direction is determined by its own curve and state, ie there is one curve for selling, and one curve for buying. This makes Carbon particularly suitable for trading applications, eg for out-of-the-money limit orders as well as for more complex trading strategies, such as "buy low, sell high" strategies operating in arbitrary user-defined ranges.
- **Adjustable parametric curves ("orders").** The parametric nature of curves (or orders, as we call them) means every user position is individually parameterized by three parameters (eg starting price, ending price and capacity) as well as one state variable (tokens held). A curve could be set to *"buy ETH for USDC between 1500 and 1000, with a total budget of USDC 100k"*, or the zero-width limit order *"sell 10 ETH for USDC at 3,000"*. Those parameters can be adjusted on the fly without closing and recreating the order, and in a non-arbitrageable manner, allowing for an easy and gas-efficient means of reacting to changes in market conditions.
- **Linked curves ("strategies").** Linked curves (or strategies, as we call them) means that multiple curves share a single collateral pool where collateral acquired on one curve is available for sale on another one; this allows for the design of advanced strategies like *"buy ETH against USDC at 1000, sell at 2000; start with USDC 10k"*.

Carbon orders and strategies can be created permissionlessly by everyone, for every possible pair of standard ERC20 tokens. This effectively creates a fully decentralized order book on-chain, against which everyone can trade. Because of the unidirectional nature of Carbon trades, they are not susceptible to key MEV attack vectors. Also there is no Impermanent Loss, in the sense that Carbon positions and strategies are not buy-and-hold liquidity positions but the expression of a particular trading view.

The Carbon protocol is enabled and supported by sophisticated matching, routing, and arbitrage algorithms that will be made open-source in due course. Especially on the arbitrage side, we also encourage others to privately identify and execute profitable intra-Carbon and Carbon versus the market arbitrage opportunities, and the authors would be very glad to engage with anyone on this particular topic, or any Carbon-related topic for that matter.

Appendix

A Formulas

In this appendix we are summarizing and briefly discussing the key formulas of the Carbon model. There is also a Jupyter Notebook that is held roughly in sync with this appendix in terms of the formulas contained, and that some readers may find useful.

A.1 Fundamental model parameters

The Carbon AMM framework is based upon five parameters, only four of which can be independently chosen as they are tied together by the invariant curve. As in the remainder of the paper, quantities denoted with x are risk asset quantities, and those denoted with y are numeraire quantities. Unless otherwise noted, all prices are quoted in units of numeraire per risk asset, ie y/x.

The "**risk asset parametrization**" of the model consist of the following parameters

- **Pool Reference Risk Asset Volume** x_0. The risk asset volume x_0 is a nominal and invariant indicator for the size of the pool, and it is denoted in units of the risk asset. It can be thought of as the virtual volume in a specific reference state.
- **Pool Reference Price** P_0. The reference price P_0 (also sometimes referred to as P) is a price-like invariant parameter that is to be read in conjunction with x_0, and that is denoted in y per x. It can be thought of as the price in the aforementioned reference state.
- **Pool Convexity** Γ. The pool convexity Γ is a dimensionless parameter that indicates the convexity of the curve. Its useful range is the interval $[0, 1]$. At $\Gamma = 0$ the curve is straight, corresponding to a constant-price AMM, and at $\Gamma = 1$ corresponds to the constant-product AMM.
- **Virtual Risk Asset Amount** x. The virtual risk asset amount x describes the current virtual state of the pool, determining its operations unless locked. The

actual state can differ due to leverage applied.

- **Virtual Numeraire Amount** y. The virtual numeraire amount y describes the current virtual state of the pool, determining its operations unless locked. The actual state can differ due to leverage applied.

In the "**numeraire parametrization**" we are using the **Pool Reference Numeraire Volume** y_0 instead of x_0. The reason why we need both is that in the numeraire parametrization the parameter P_0 does not affect the pool size when looked at in numeraire terms, but it does affect it when looked at in risk asset terms, and vice versa. We also have the "**volume parametrization**" that uses x_0, y_0 instead of P_0 but this is more rarely used.

Within the risk asset parametrization there are two particularly interesting alternatives which both keep the ratio x, x_0 fixed. Those are the **r-parametrization** based on

$$x_0 = rx \tag{A.1}$$

and the **rho-parametrization** using

$$x = \rho x_0 \tag{A.2}$$

A.2 Invariant curves

As already alluded to above, the **pool invariant curves** tie together the five parameters and state variables of the pool, reducing them effectively to 3 parameters and one state. In the **risk asset parametrization** x_0, P_0, Γ the invariance formula looks as follows

(desmos)

Invariant Equation as function of Γ, P_0, x_0

$$y = \frac{P_0 x_0 \left(x \left(\Gamma - 1\right) - x_0 \left(\Gamma - 2\right)\right)}{\Gamma x - x_0 \left(\Gamma - 1\right)} \tag{A.3}$$

rewritten using r

$$y = -\frac{P_0 r x \left(-\Gamma + r \left(\Gamma - 2\right) + 1\right)}{\Gamma - r \left(\Gamma - 1\right)} \tag{A.4}$$

rewritten using ρ

$$y = \frac{P_0 x_0 \left(-\Gamma + \rho \left(\Gamma - 1\right) + 2\right)}{\Gamma \rho - \Gamma + 1} \tag{A.5}$$

in the **numeraire parameterization** as function of Γ, P_0, y_0

(desmos)

$$y = \frac{y_0 \left(P_0 x \left(\Gamma - 1\right) - y_0 \left(\Gamma - 2\right)\right)}{P_0 \Gamma x - y_0 \left(\Gamma - 1\right)} \tag{A.6}$$

in the **Q-form**; here Q is a dimensionless parameter equivalent to Γ, and x_{int} and y_{int} are places where the curve cuts through the x and y axis respectively. See the next section for a more thorough discussion of those parameters and for related formulas.

(desmos)

$$Q = \frac{xy}{\left(x - x_{int}\right)\left(y - y_{int}\right)} \tag{A.7}$$

In the **asymptotic form** as function of the asymptotes x_{int}, y_{int} and the levered pool constant κ

$$\left(x - x_{asym}\right)\left(y - y_{asym}\right) = \kappa \tag{A.8}$$

A.3 Swap and marginal price equations

The **swap equation** is the fundamental operational equation of the AMM, describing what happens in a transaction. In addition to the parameters, we have seen before it also has the *swap amounts* Δx (risk asset) and Δy (numeraire). Those are signed quantities, and the sign is observed from the vantage point of the AMM, so $\Delta x > 0$ means the AMM is buying the risk asset, and $\Delta x < 0$ means it is selling it. Evidently $\Delta x, \Delta y$ always have the opposing sign, hence the sign in front of the equation. Our representation of the formula isolates the numeraire change Δy as a function of the risk asset trading volume Δx, but of course the risk asset view $(-\Delta x = \cdots)$ or the effective price view $(-\Delta y/\Delta x = \cdots)$ of this equation are trivially obtained.

$$-\Delta y = \frac{P_0 \Delta x x_0^2}{\left(\Gamma\left(x - x_0\right) + x_0\right)\left(\Gamma\left(\Delta x + x - x_0\right) + x_0\right)} \tag{A.9}$$

$$\Delta y = \frac{P_0 \Delta x r^2 x}{\left(\Gamma\left(r - 1\right) - r\right)\left(\Gamma\left(\Delta x - rx + x\right) + rx\right)} \tag{A.10}$$

$$-\Delta y = \frac{P_0 \Delta x x_0}{\left(\Gamma\left(\Delta x + x_0\left(\rho - 1\right)\right) + x_0\right)\left(\Gamma\left(\rho - 1\right) + 1\right)} \tag{A.11}$$

The **marginal swap equation** is the swap equation for an infinitesimal Δx. It is here shown in the price view, where dx and dy can either be interpreted as infinitesimal quantities, or dy/dx can be interpreted as both the marginal price and the derivative of the invariance function.

(desmos)

$$-\frac{dy}{dx} = \frac{P_0 x_0^2}{\left(\Gamma\left(x - x_0\right) + x_0\right)^2} \tag{A.12}$$

The **marginal price equation** is exactly the same as the marginal swap equation, except that we rename dy/dx as P_{marg} to honor the fact that (a) this quantity is the marginal price, and (b) arguably this is the most important amongst all the equations presented.

(desmos)

Marginal Price Equation as function of Γ, P_0, x_0

$$P_{marg} = \frac{P_0 x_0^2}{\left(\Gamma\left(x - x_0\right) + x_0\right)^2} \tag{A.13}$$

rewritten using r

$$P_{marg} = \frac{P_0 r^2}{\left(\Gamma\left(r - 1\right) - r\right)^2} \tag{A.14}$$

rewritten using ρ

$$P_{marg} = \frac{P_0}{\left(\Gamma\left(\rho - 1\right) + 1\right)^2} \tag{A.15}$$

as function of Q, P_0, x_{int} ($P_{\dots} \simeq y/x$)

$$P_{marg} = \frac{P_0 Q x_{int}^2}{\left(-Qx + Qx_{int} + x\right)^2} \tag{A.16}$$

as function of Q, P_0, y_{int} ($P_{\dots} \simeq y/x$)

$$P_{marg} = \frac{P_0\left(-Qy + Qy_{int} + y\right)^2}{Q y_{int}^2} \tag{A.17}$$

as function of w, P_0, y_{int} where $P_{\dots} \simeq y/x$ and $\sqrt{1/w} = Q$

$$P_{marg} = \frac{P_0 \left(-y\sqrt{\frac{1}{w}} + y + y_{int}\sqrt{\frac{1}{w}}\right)^2}{y_{int}^2 \sqrt{\frac{1}{w}}} \tag{A.18}$$

A.4 Parameters and their relationships

The first parameter we look at is the **Q-value** Q that we have already seen above. In the interval $[0, 1]$, which is the one we focus on, Q is equivalent to Γ. Note that direction is reversed however, and $Q = 0$ is the constant-product case, whilst $Q = 1$ is the constant-price case.

(desmos)

$$Q = (1 - \Gamma)^2 \tag{A.19}$$

the inverse relationship is of course this one

(desmos)

$$\Gamma = 1 - \sqrt{Q} \tag{A.20}$$

Also, we have the relationship below, which is useful in the equations that follow

(desmos) (desmos)

$$\frac{2 - \Gamma}{1 - \Gamma} = 1 + \frac{1}{\sqrt{Q}} \tag{A.21}$$

The **risk asset intercept** x_{int} is the point at which the invariant curve intercepts the x-axis, and it is denominated in terms of the risk asset. Unsurprisingly it is proportional to x_0, and the factor only depends on Γ. For $\Gamma = 1$, the constant-product case, this quantity diverges as the curve is asymptotic to the axis. The slope of the intercept is P_x.

(desmos) (desmos) (desmos)

$$x_{int} = \frac{x_0 (2 - \Gamma)}{1 - \Gamma} \tag{A.22}$$

The **numeraire intercept** y_{int} is analogous to x_{int}, except that it is against the y-axis, and is therefore denominated in the numeraire.

(desmos) (desmos) (desmos)

$$y_{int} = \frac{y_0 (2 - \Gamma)}{1 - \Gamma} \tag{A.23}$$

The **risk asset intercept price** P_x is the slope of the invariant curve at the point of intercepting the x-axis, x_{int}. It is price-like, denoted in y/x. We remind ourselves that $Q = 1$ is constant-price, in which case $P_x = P_0$. Also, $Q = 0$ is constant-product where there is no intercept, so the price goes formally to 0.

(desmos)

$$P_x = P_0 Q \tag{A.24}$$

The **numeraire intercept price** P_y is the slope of the invariant curve at the point of intercepting the y-axis, y_{int}. Despite being on the other axis it is price-like, denoted in y/x, just like P_x. For $Q = 1$, the constant-price case, we find $P_x = P_0$. For $Q = 0$, the constant-product case where there is no intercept, the price diverges.

(desmos)

$$P_y = \frac{P_0}{Q} \tag{A.25}$$

We already mentioned the fundamental equation tying P_0, x_0, y_0 at the beginning of the appendix but here it is again for reference.

$$P_0 = \frac{y_0}{x_0} \tag{A.26}$$

As x_{int}, y_{int} are proportional to x_0, y_0 with the same factor, the reference price P_0 can also be recovered from the intercepts. This relationship is interesting from a financial point of view: y_{int}/x_{int} can be interpreted as the effective price of a series of limit order trades, and this equation shows that the average price at which they are filled does not depend on Γ.

(desmos)

$$P_0 = \frac{y_{int}}{x_{int}} \tag{A.27}$$

The relationship between Q and the intercept prices is trivial from the definition of those prices, but it is interesting to note, nevertheless. If $Q = 1$ (constant-price) then the curve is a line and therefore the two prices are the same. For Q getting smaller the convexity increases, meaning the P_y becomes bigger (steeper) and P_x smaller (flatter).

(desmos)

$$Q = \sqrt{\frac{P_x}{P_y}} \tag{A.28}$$

This relationship is even more interesting in that form, linking the range width to Q^2

$$w = \frac{1}{Q^2} \tag{A.29}$$

Again, this relationship between the pool reference price P_0 and the intercept prices P_x, P_y is trivial from their definitions, but it is worth noting, nevertheless. It also echoes the fact that the effective trading price of an AMM is the geometric average of the prices at the trade boundaries.

(desmos)

$$P_0 = \sqrt{P_x P_y} \tag{A.30}$$

The **asymptotes** x_{asym}, y_{asym} are a measure of the x and y translation of the curve in the respective direction. Those are non-positive numbers, meaning the curve has been moved to the left and down. In terms of our other paramters, the x-asymptote is

$$x_{asym} = x_0 \left(1 - \frac{1}{\Gamma}\right) \tag{A.31}$$

and the y-asymptote is

$$y_{asym} = P_0 x_0 \left(1 - \frac{1}{\Gamma}\right) \tag{A.32}$$

The most important use of those parameters is the *asymptotic pool invaration equation* above, where they are combined with the **levered pool constant**

$$\kappa = \frac{P_0 x_0^2}{\Gamma^2} \tag{A.33}$$

When reminding ourselves that $P_0 x_0^2 = x_0 y_0 = k$, the important corollary of the abpve equation is that as $\sqrt{k}$ is a measure of the pool size, $1/\Gamma$ is the leverage factor, ie a measure for the amplification of the leverage in the pool due to the narrowing of the range.

B Implementation formula reference

In this appendix, we focus on formulas that are important for the implementation. We will provide very little commentary here – this appendix is meant as a formula reference for people already familiar with the model, not as an introduction.

Prices are always quoted in y per x; for historical reasons we use multiple names for the same prices: x, y denotes the axis intercept, to which they relate, a, b is reading them left to right, and min, max is ordering them in their natural quotation ie $P_{max} > P_{min}$.

$$\begin{aligned} P_a &= P_y = P_{max} \\ P_b &= P_x = P_{min} \end{aligned} \tag{B.1}$$

The key implemenation parameters are B, S

$$\begin{aligned} B &= \sqrt{P_b} \\ S &= \sqrt{P_a} - \sqrt{P_b} \\ P_b &= B^2 \\ P_a &= (S+B)^2 \end{aligned} \tag{B.2}$$

Alternatively we can define S based on $P_0 = \sqrt{P_a P_b}$ and the shape parameters Q or Γ as

$$S = \sqrt{\frac{P_0}{Q}}\,(1-Q) = \sqrt{P_0}\,\frac{\Gamma\,(2-\Gamma)}{1-\Gamma} \tag{B.3}$$

B, S satisfy the following relationships

$$\begin{aligned} B^2 &= P_b \\ S^2 &= P_a + P_b - 2\sqrt{P_a\,P_b} \\ BS &= \sqrt{P_a\,P_b} - P_b = P_0 - P_b \\ B^2 + BS &= \sqrt{P_a\,P_b} = P_0 \\ S^2 + BS &= P_a - \sqrt{P_a\,P_b} = P_a - P_0 \end{aligned} \tag{B.4}$$

The curve capacity is y_{int}, and its utilisation is y; the utilization ratio is closely related to the marginal price

$$\begin{aligned} \frac{y}{y_{int}} &= \frac{\sqrt{P_m} - \sqrt{P_b}}{\sqrt{P_a} - \sqrt{P_b}} = \frac{\sqrt{P_m} - B}{S} \\ P_m &= \left(S \cdot \frac{y}{y_{int}} + B\right)^2 \end{aligned} \tag{B.5}$$

Effective prices and marginal prices, and matching by source

$$\begin{aligned} -\frac{\Delta y}{\Delta x} &= \frac{(Sy + By_{int})^2}{y_{int}^2 + \Delta x S(Sy + By_{int})} \\ P_m = -\frac{dy}{dx} &= \frac{(Sy + By_{int})^2}{y_{int}^2} = \left(S \cdot \frac{y}{y_{int}} + B\right)^2 \end{aligned} \tag{B.6}$$

Matching by target

$$-\frac{\Delta x}{\Delta y} = \frac{y_{int}^2}{(Sy + By_{int})(Sy + By_{int} - S\Delta y)} \tag{B.7}$$

Invariant functions

$$\begin{aligned} y &= \frac{y_{int}(y_{int} - x(B^2 + BS))}{x(BS + S^2) + y_{int}} = \frac{y_{int}(y_{int} - x\sqrt{P_a\,P_b})}{x(P_a - \sqrt{P_a\,P_b}) + y_{int}} \\ x &= \frac{y_{int}(y_{int} - y)}{y_{int}(B^2 + BS) + y(BS + S^2)} = \frac{y_{int}(y_{int} - y)}{y_{int}\sqrt{P_a\,P_b} + y(P_a - \sqrt{P_a\,P_b})} \end{aligned} \tag{B.8}$$

Curve capacities in the respective tokens

$$x_{int} = \frac{y_{int}}{B^2 + BS} = \frac{y_{int}}{\sqrt{P_a P_b}} \tag{B.9}$$

C Resources

- The Carbon website is at carbondefi.xyz
- On the Carbon website there is the current whitepaper, our litepaper, the innovation disclosure relating to our patent application, and a mathematical summary
- On the Bancor github there is the Carbon Simulator. The simulator repo contains
  - a number of example workbooks looking at Carbon Orders (Demo1-1, Demo1-2, Demo1-3), Carbon Strategies (Demo2-1, Demo2-2, Demo2-3) and the resulting order books (Demo3-1, Demo3-2),
  - the example workbooks for the whitepaper CarbonSim-WhitepaperExample), and the litepaper CarbonSim-LitepaperExample and CarbonSim-LitepaperExampleShort),
  - and the workbook providing the key Carbon formulas and calculations in SymPy, CarbonCalculations.