Shell Protocol White Paper

October 1, 2020

Abstract

The objective of the Shell Protocol is to create an internet monetary system using stablecoins as building blocks. The first release is a liquidity pool optimized for stablecoin-to-stablecoin trades. It has weights, deep liquidity, protections against a broken peg and dynamic fees. It can also provide liquidity directly between stablecoin derivatives such as cTokens and aTokens. The goal for this iteration is to develop a framework for flexible bonding curves that can adapt to new use cases and market conditions. Liquidity provider shares are called "shells" because they are containers for the value held by the pool, just as living shells are containers for the organism inside. Shells are natively liquid, diversify exposure and earn yield. They have the potential to become the primary means of storing and transacting value.

1 Internet money

The Shell Protocol's purpose is to create an internet monetary system: a borderless, programmable medium of exchange accessible to all. This dream was the original promise of Bitcoin.[21] Because of its static supply, Bitcoin is too volatile to be used as money. Somehow, in the past few years, the mission has fallen by the wayside. But we should not give up. This white paper will explain how we can create internet money using stablecoins as building blocks.

Banks are the foundation of legacy money; stablecoins are the foundation of internet money. The banking system is an ad hoc confederation of private ledgers replete with redundancy and a commensurate level of inefficiency. Stablecoins exist on public blockchains, are internet native and inherently programmable. It is inevitable that stablecoins will supersede banks, probably a lot faster than we expect. US regulators are already starting to embrace USD stablecoins.[11]

No single stablecoin will constitute internet money on its own. Instead, it will take an entire ecosystem of stablecoins. A healthy stablecoin ecosystem needs a diverse and integrated market. I.e., there needs to be many different kinds of stablecoins and transferring value between them needs to be seamless. However, the current stablecoin market is highly concentrated and fragmented. Indeed, stablecoins are more concentrated than the US banking system. The top five US banks are roughly equivalent in size and constitute no more than 40% of the total value

held by the system (see Figure 1). In contrast, the top five stablecoins are highly unequal, with the number one stablecoin, Tether[27], holding roughly eight times the value of the number two stablecoin, USD Coin[5]. Collectively, the top five stablecoins constitute over 95% of the market (see Figure 2).

It is a great irony that "decentralized" finance (DeFi) is more centralized than legacy finance. We do not want stablecoins to be too big to fail. We need a protocol that can unify stablecoins into a coherent, yet diverse, ecosystem. Stablecoin liquidity pools are an important first step.

A liquidity pool allows people, called "liquidity providers", to contribute their capital to an autonomous system that will use this capital to facilitate trades between the reserve assets held by the pool. The price of these trades are determined by an automated market maker (AMM).[4] For example, if Alice contributes A tokens to a pool and Bob contributes B tokens to a pool, then the pool can now facilitate trades between A and B, according to its internal logic.

The power of a liquidity pool built on smart contracts is that it solves the social scalability problem[26] for liquidity. Because the system can operate autonomously, liquidity providers like Alice and Bob do not need to know or trust each other, nor do they need to know or trust a professional market maker. They only need to have confidence in the AMM contract, which they can independently verify. This unlocks large amounts of otherwise idle capital that can now be used to create liquidity between assets. In effect, pools charge a liquidity premium to the rest of the market while taking on volatility risk. Inter-stablecoin pools are especially effective because as long as stablecoins do not permanently lose their peg, the volatility risk is relatively low.

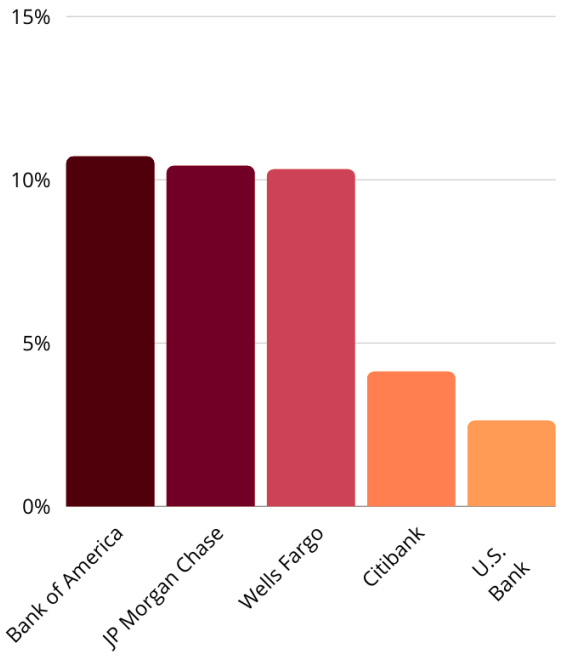

Figure 1: Market share of top five US banks in terms of deposits [24]

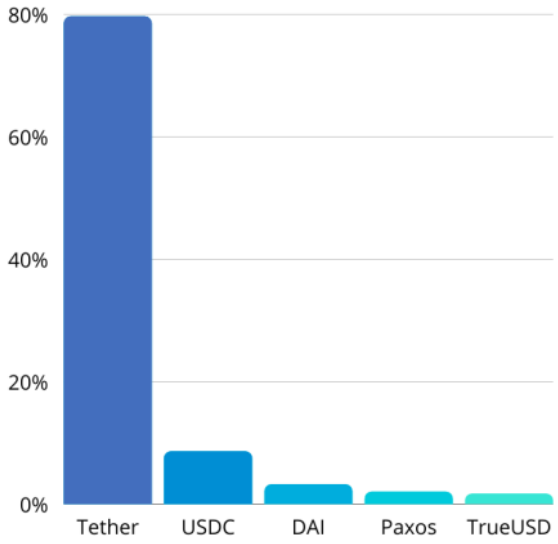

Figure 2: Market share of top five stablecoins in terms of market cap [7]

The current market leaders for stablecoin liquidity pools are Curve[13] and Swerve[25]. At the time of writing (September, 2020), they each have $1.2 billion[9] and $41 million[25] worth of stablecoins in their pools respectively. And they processed $98 million and $2.1 million in daily trade volume. Less than four months prior in June, 2020, Curve had only $16 million in stablecoins and Swerve did not even exist. Although the rate of growth has been impressive thus far, we are still very early in the transition cycle. As an analogy, Fedwire, the payment rails for the US banking system, processes roughly $750 trillion per year.[14] Stablecoins will grow to at least this size, if not orders of magnitude larger. The world sends far more emails today than letters through the postal service. Inter-stablecoin volume versus Fedwire volume will follow the same pattern.

Because stablecoins are the seed which will grow into the future internet monetary system, we need to think beyond just AMMs and liquidity pools. Beyond even singular stablecoins. Instead, we must think about shells.

2 What is a shell?

Recall that stablecoins in pools are contributed by liquidity providers. In order to track contributions, the pool issues shares to whomever deposits tokens. To withdraw their tokens from the pool, liquidity providers redeem their shares. We call shares in a liquidity pool "shells" because they are a container for the underlying assets in the pool, just as a living shell is a container for the organism inside.

Shells have three primary advantages over inert stablecoins:

1. Liquidity
2. Diversification
3. Yield

In effect, shells are tokenized liquidity pools. As such, they inherit the very liquidity utilized for swaps. Stablecoins in the pool can directly convert to shells, and vice versa shells can directly convert to stablecoins. In that sense, shells are natively liquid assets.

Shells offer more than just an efficient source of liquidity. They are diversified portfolios of assets. Diversification can lower the risk of holding shells vs. a single stablecoin. The likelihood of one stablecoin losing its peg is higher than the likelihood of all stablecoins losing their peg. Additionally, the liquidity pool can earn passive income on behalf of shell holders by providing liquidity to the broader market. For every swap, a fee is earned by the pool which accrues directly to the shells, earning passive income. Hence, shells are yield bearing assets.

Given that shell tokens are diversified, inherently liquid and earn yield, they have the potential to be the best store of value of any crypto asset. Moreover, it is also easier to manage a single shell than multiple stablecoins. In the legacy financial system, the safest and most liquid interest bearing assets are US Treasuries. Currently, no asset fulfills that niche in crypto markets.

2.1 A protocol for shells

If and when shells become the default means of exchanging and storing value (i.e. "money"), what will such a system look like? Although pegged to the same numeraire, stablecoins will still have meaningful differences such as different risk profiles, regulatory constraints, geographic distributions, etc. For example, stablecoins can be backed by fiat currency held in an audited bank account like USD Coin or Paxos[23]. Or fiat reserves can be held in unaudited accounts like Tether. Maker[18] and Synthetix[6] each issue their own stablecoin backed by cryptocurrency. Terra has a stablecoin whose peg is backed by future cash flows generated from transaction fees.[16] The variety of stablecoins will only increase over time. Moreover, as stablecoins gain broader adoption, users will become increasingly heterogeneous. Early DeFi adopters have a high risk tolerance but this will not be the case for mainstream users.

Will the entire world use the same stablecoin shell? No, of course not. What stablecoins would go into that shell? What weights will the reserve assets have? How risky should the pool be? Agreeing on a single shell that satisfies everyone's needs will be impossible. There will be myriad pools, thousands of shells. Perhaps just as importantly market conditions are highly variable. A pool design that works well when stablecoin prices are stable will not be optimal when stablecoin prices are volatile (see Section 4.2).

Therefore, a protocol for creating shells will need to be highly configurable to adapt to the many different stablecoins and market conditions. This paper will describe and formalize our model for creating shells and liquidity pools. The emphasis for this first iteration is to create a

framework for highly flexible core logic. That is, by merely tuning parameters, the behavior of the pool can be meaningfully altered.

Shell is open source so anyone could technically deploy a pool without needing permission. However, given the novelty of the protocol, independent deployments will not be encouraged or supported initially until the project matures. Ultimately, it will become easy for people to configure, deploy and operate their own pools.

In principle, the model presented can accommodate any intra-numeraire liquidity pool, not just stablecoins. For example, a BTC-to-BTC pool will be just as viable as a USD-to-USD pool. From the standpoint of the core logic, "stable" refers to the relative value of reserves, not their aboslute value. Given Shell's mission, the focus of this paper will be on USD stablecoins.

The rest of the paper will be structured as follows. **Section 3** will give a high-level description of the protocol. **Section 4** will visualize the model via bonding curves. **Section 5** will discuss initial use cases for Shell. **Section 6** will formalize Shell's AMM model. In particular, **Section 6.1** rigorously defines key terminology. Lastly **Section 7** will layout the protocol's future direction.

3 Description of Shell Protocol

There are five dimensions along which Shell liquidity pools can be configured:

1. Ideal weights for reserve stablecoins (w)
2. Depth of 1:1 exchange (β)
3. Price slippage (elasticity) when not 1:1 exchange (Δ)
4. Maximum and minimum allocation of each reserve (α)
5. Fees (ϵ, λ)

3.1 Reserve weights

Each stablecoin reserve in the pool has an ideal weight, w. It determines what percentage of each reserve the pool wants to hold. If $w = 0.5$, then the ideal weight for that reserve will be 50% of the total reserves held by the pool. The first Shell pool deployed will be 30% DAI, 30% USDC, 30% USDT and 10% SUSD. When a pool is "balanced", actual weights are close to ideal weights, the pool functions as a constant-sum market maker. The sum of all the balances remains unchanged before and after a swap. Exchange between stablecoins will be 1:1 and there will be no slippage except for a fixed fee, ϵ (see Section 3.4) .

3.2 Liquidity depth and price slippage

The ideal weight interacts with another parameter, β, to determine the liquidity depth for that reserve. Once the actual weight is more than β percent away from the ideal weight, transactions

will begin to incur slippage. For example, if $w = 0.5$ and $\beta = 0.75$, then the token will encounter increasingly large slippage if it is lower than 12.5% or larger than 87.5% of the pool. Therefore, the higher the value of β, the deeper the pool. The rate at which price slippage increases (elasticity) once past β is determined by the parameter Δ. The higher the value of Δ, the more rapidly slippage will increase. If $\Delta = 0$, then there will be no price slippage at any point. While this may seem attractive, without price slippage even a small deviation from the peg will drain the pool of all its liquidity. Thus, no price slippage will actually make the pool less liquid, especially for stablecoins that deviate above the price peg. This exact phenomenon happened to mStable, a "meta" stablecoin basket with a no-slippage AMM.[20] One of the reserve stablecoins, DAI, typically trades above its peg, and the mStable pool has a chronic DAI shortage.

3.3 Protection from a broken peg

To guard against a broken stablecoin peg, the pool has a minimum and maximum allocation for each reserve. This allocation is determined by α. The higher the value of α, the lower the minimum allocation percent and the higher the maximum allocation percent. If the ideal weight for the reserve is $w = 0.25$ and $\alpha = 0.8$, then the pool will not accept any transaction that increases the actual weight of the reserve beyond 45%. Without this constraint, a broken peg would drain the entire pool as traders will continue to add the worthless stablecoin and remove the viable stablecoins until there is no value left. Without setting a threshold for each stablecoin, then diversifying reserve assets will make a pool riskier because it only takes one broken peg to completely drain the pool. This is the case with AMM protocols such as Curve, Uniswap[2] and Balancer[19]. Because Shell pools have a minimum and maximum allocation parameter, diversification can actually reduce, rather than increase, risk.

3.4 Yield

The pool earns yield for shells by charging a fixed fee, ϵ and a dynamic fee, λ. Every time value leaves the pool, ϵ percent of that value is retained, accruing directly to shells. In addition, the pool also charges a dynamic fee on any transaction in the price slippage zone. In other words, the more a stablecoin deviates from its price peg, the higher the bid/ask spread becomes. As discussed, any transaction that takes a reserve too far past its ideal weight will incur an unbalancing penalty. Conversely, any transaction that brings a reserve closer to its ideal weight when starting in the slippage zone will be awarded a rebalancing subsidy. Unbalancing the pool incurs slippage; rebalancing a pool incurs anti-slippage. A parameter, λ, determines the difference between the unbalancing fee and the rebalancing subsidy. The difference, $(1 - \lambda)$, is retained by the pool as a dynamic fee. If $\lambda = 1$, then there is no dynamic fee. If $\lambda = 0.5$, then the dynamic fee will be half the price slippage.

Dynamic fees can be especially lucrative for the pool when there is price volatility. Every time a stablecoin deviates from its peg, the pool will capture a portion of this slippage. Suppose $\lambda = 0.7$ and $\epsilon = 0.00035$ (3.5 basis points). If prices deviate by 5% from the peg, then the pool would earn roughly 100x more from the dynamic fee than from the fixed fee. The trade

off of dynamic fees is that they reduce the incentive for arbitrage traders to interact with the pool, thus reducing trade volume. However, in the example above, trade volume would have to decrease more than one hundred fold to offset the income generated from the dynamic fee. Further modeling will be needed to infer the ideal dynamic fee, which will also depend on market conditions and stablecoin reserves.

3.5 Dynamic parameters

All of these parameters (α, β, Δ, ϵ, λ) can be adjusted post deployment in order for the pool to dynamically adapt to new market conditions. The pool can have deep 1:1 liquidity when there is low volatility and then readjust parameters when there is high volatility (see Section 4.2). For the first release, weights will be fixed at deployment but subsequent iterations will enable dynamic weights. The only constraint on updating parameters is that the value held by the pool (as measured by its utility function, see Section 6.3) cannot decrease as a result of the new parameters. That way, a centralized pool operator cannot remove funds from the pool. The Shell model can work for more than just stablecoins. As long as all reserves are denominated in the same numeraire (e.g. USD, BTC or ETH), then the model works equally well. Thus a pool could create liquidity between BTC on Ethereum or various staking derivatives.

4 Shell visualized

It is helpful to visualize the Shell Protocol and see how adjusting parameters can affect behavior. First, let's explain how we can visualize an AMM's logic. Imagine a pool has two reserve assets with balances: $[x, y]$. If someone wants to exchange $\hat{x}$ for $\hat{y}$, then the pool's new reserve balances will be $[x+\hat{x}, y-\hat{y}]$. We can graph all possible values of $[x+\hat{x}, y-\hat{y}]$ on an XY-plane. We call this plot a "bonding curve"[10] and embedded within it is the behavior of the pool's AMM. Because Shell's AMM uses a utility function (see Section 6.3), the bonding curve is also an indifference curve.[15] The slope of the bonding curve at any given point (i.e. the derivative) is the spot price of y in terms of x.

In this section, we will visualize three examples:

1. "Standard" Shell bonding curve (Figure 3)
2. Overlaid bonding curves with different parameter sets (Figure 4)
3. Bonding curve with dynamic fees (Figure 5)

Each example will feature evenly weighted pools ($w_x = w_y = 0.5$) with reserve balances of $[x, y] = [50, 50]$ when at the ideal weights.

4.1 "Standard" bonding curve

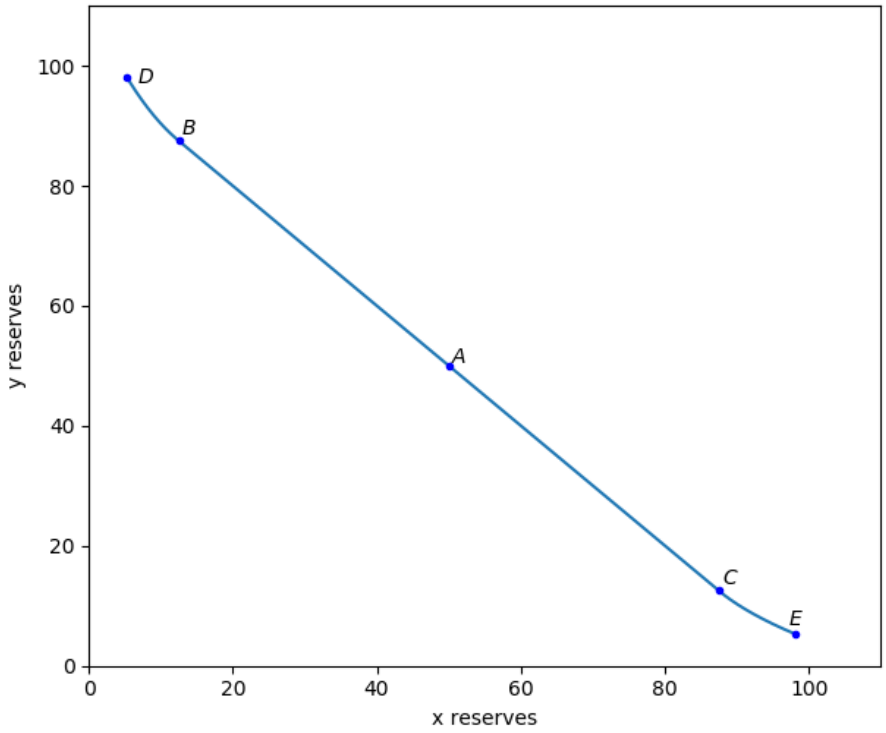

Figure 3: Bonding curve for an evenly balanced pool.

Figure 3 demonstrates a "standard" Shell bonding curve with deep liquidity. At point A, the pool is perfectly balanced. Points B and C are the beginning of the slippage zone with $\beta = 0.75$. Hence, the length of segment BC is the depth of the pool. The rate of price change in the slippage zone is determined by $\Delta = 1.5$. Points D and E are the halt boundary with $\alpha = 0.9$.

Recall that the slope of the graph (derivative) determines the price of y in terms of x. The region where the line is straight with a slope of $\frac{\partial y}{\partial x} = -1$ (BC) the exchange is 1:1. That is, there is no price slippage. When the pool is overweight x and underweight y, the graph curves horizontally (CE), and the price of x will decreases relative to y. Conversely, when the pool is underweight x and overweight y, the graph curves vertically (DB), and the price of x will increase relative to y.

The important thing to realize about this graph is that each point (A, B, C, D and E) can be altered merely by changing the parameters. Thus the AMM can adapt its behavior to different market conditions (see Section 4.2).

4.2 Flexible curves

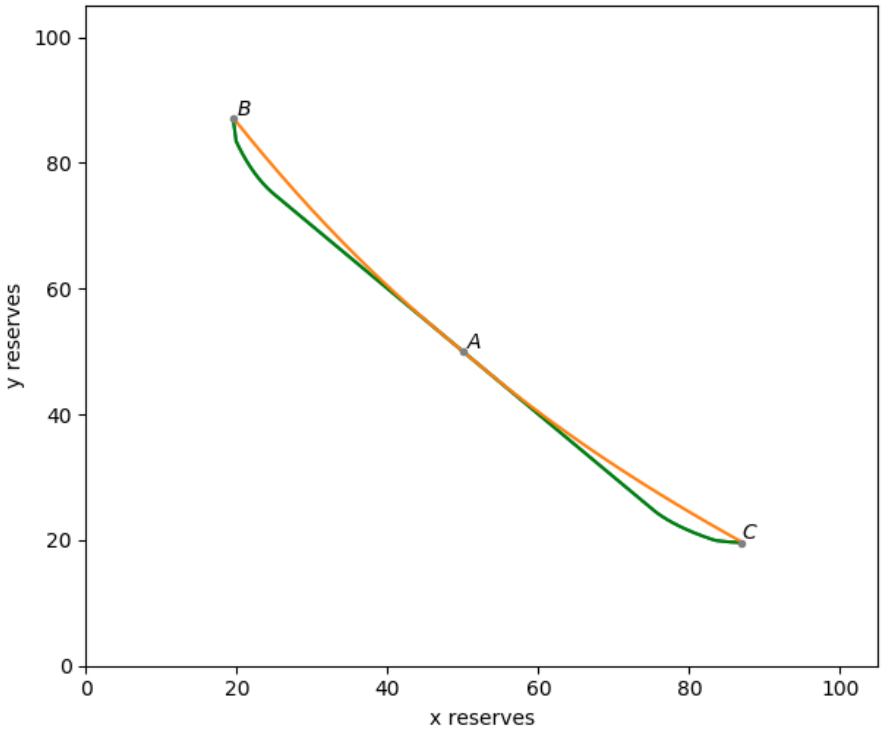

Figure 4: Two bonding curves with the same balance point but different parameters.

To demonstrate how different parameters can influence the shape of the bonding curve (and hence the behavior of the model), we can overlay two curves onto one graph in Figure 4. They both share the same balance point of A. However, $curve_1$ (green) has a shallower no-slippage zone than $curve_2$ (orange) with $\beta_1 = 0.5$ and $\beta_2 = 0.05$. However, $curve_1$ has a lower slippage parameter than $curve_2$ with $\Delta_1 = 2.5$ and $\Delta_2 = 0.185$. Thus, both curves end up at the same halting points of B and C.

Recall that when a stablecoin deviates from its peg, arbitrage traders will trade against the pool until they reach the slippage zone (see Section 3.2). If a curve has a wide segment for 1:1 exchange (BC in Figure 3) it will lose more of its liquidity when there is high price volatility. If a curve has a narrow segment for 1:1 exchange and more curvature, it will lose less of its liquidity when the market is volatile. Therefore, $curve_2$ would be better suited to markets where stablecoins have relatively high volatility. Whereas $curve_1$ would be better suited to markets where stablecoins have relatively low volatility. The flexibility of Shell's model means that the same pool can morph its bonding curve according to the market, selecting the optimal trading

strategy.

4.3 Dynamic fee

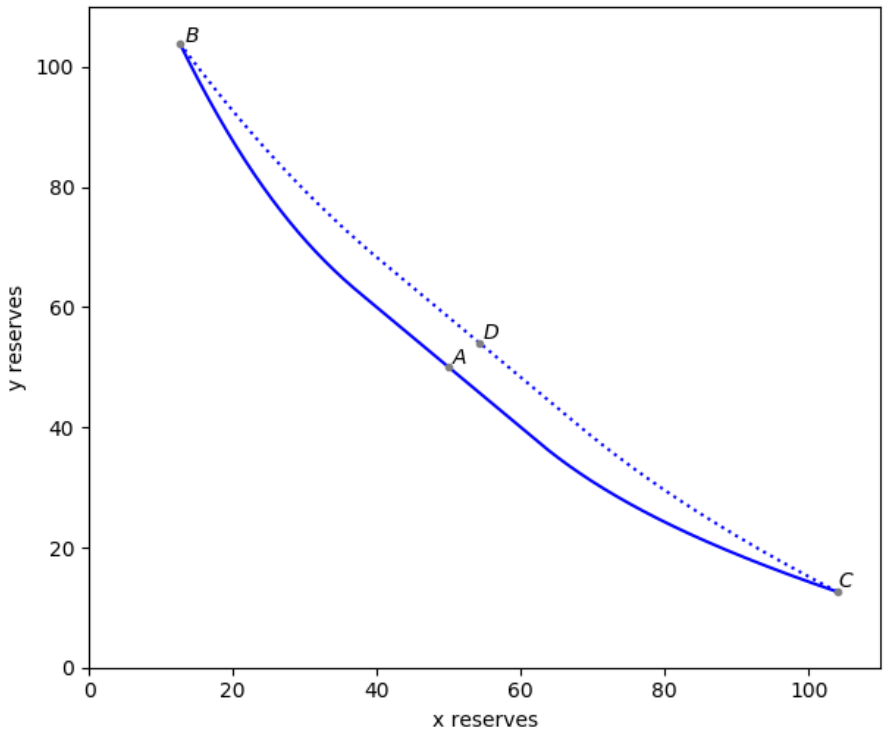

Figure 5: Dynamic fees represented by the dotted line

How can we visualize the dynamic fee? In Figure 5, we start out at the balance point, A. Then we swap all the way to either halt thresholds, B or C (solid line). If we reverse the swap and return to the balance point, the dynamic fee will be assessed by setting the anti-slippage to be worth λ of the slippage. In this case, $\lambda = 0.5$, so anti-slippage is half of the slippage. Therefore, when we return from the halt boundaries back to the new balance point, $D = [54, 54]$, the pool's reserves will be higher than before (dotted line). The distance between A and D is the dynamic fee captured by the liquidity pool and accrued to the shell tokens.

Without the dynamic fee, the income earned moving from $A \to B \to C$ would be approximately 0.0378 (versus starting reserves of 100). With the dynamic fee, the total income earned was 8. Hence, the dynamic fee in this example generated over 200x more income for the pool than the fixed fee. And like the shape of the bonding curve, the dynamic fee can be adjusted to adapt to market conditions.

5 Use cases

The Shell Protocol's most obvious use case is as a source of liquidity for DeFi traders and a source of yield for liquidity providers. Because liquidity begets trade volume, liquidity providers are the initial focus. Given that Shell has dynamic fees and protections against a broken stablecoin peg, it has the potential to earn higher yield for less risk versus incumbent stablecoin AMMs that have no protections and only a constant fee. Shell pools can also adjust dynamically to market conditions.

But the real advantage of Shell for liquidity providers lies in the model's flexibility. For example, stablecoin reserves can have precise weights in the pool, which is unique compared to other stablecoin AMMs. Thus, relatively low volume stablecoins, such as SUSD, can have a smaller weight versus high volume stablecoins such as Tether. Indeed, this is exactly the structure of the first Shell pool, with 10% allocation for SUSD and 30% allocation for Tether.

5.1 Bootstrapping liquidity for new stablecoins

Weights can also be used in reverse, to bootstrap liquidity for a new stablecoins. One could create a pool with 50% target weight allocated to the new stablecoin. This pool can then receive liquidity mining incentives[3] from the new stablecoin's parent protocol. With an evenly weighted pool, liquidity incentives will be diluted between all other stablecoins in the pool, whereas an overweight pool will generate more liquidity per dollar of subsidy.

5.2 Liquidity between stablecoin derivatives

Shell is not limited to promoting liquidity between stablecoins, staking derivatives or bitcoin on Ethereum. Shell can also provide liquidity between lending pools and smart yield protocols such as Compound[17], Aave[1] and yEarn[29]. For example, you could directly swap between cUSDC to aUSDT in one transaction without using a Zap[30]. The same flexibility applies to the pool's reserves. Stablecoins can be deposited into yEarn and earn additional yield on top of trading fees from the AMM. The architecture is such that more integrations can be added without needing to change the pool. For security purposes, once a pool is deployed, new integrations will not be added to that pool. As governance of the pools is decentralized, then this constraint can be relaxed.

5.3 Settlement currency for DeFi protocols

The protocol offers more than just liquidity and yield. Shell tokens can be a direct medium of exchange for other DeFi protocols. Recall that shells can convert to stablecoin reserves, and vice versa stablecoin reserves can convert to shells (see Section 2). For example, if you are a decentralized options exchange, such as Opyn[22], your market must settle in a stablecoin. Ideally, the exchange can be stablecoin agnostic by accepting many different stablecoins without fragmenting liquidity.

Instead of settling via a single stablecoin, trades can settle in shell tokens. Suppose Alice has sold Bob a put option. Alice wants to settle the option in DAI while Bob wants to settle the option in USDC. At the smart contract level, the option can settle in shells. If Bob exercises the option, the shells will be converted to USDC. If the option is not exercised, the shells will be converted to DAI and returned to Alice.

Given that shells can convert between lending pools and other stablecoin derivatives, DeFi protocols can accept a vast array of assets on their platform without needing to add extra integrations. Moreover, they can create their own pool with parameters customized for their needs. Using other stablecoin AMMs in this way is extremely risky if they do not have protections against a broken peg (see Section 3.3). Without these protections, a single broken peg will drain the entire pool and take any other DeFi exchange down with it.

6 The Shell protocol formal model

We are now ready to formalize the Shell Protocol model for liquidity pools. The formalism will first cover the key terms, behaviors and parameters of pools. Then it will explain how the protocol computes deposits, withdrawals and swaps. For a less mathematically driven explanation of how the model works, see Section 3. To visualize Shell's model, see section 4.

6.1 Definitions

Before going any further, it is necessary to define with rigor the key terminology. For expediency, this information will be presented as a list:

Liquidity Pool: A smart contract that holds and manages several different kinds of fungible assets, such as ERC-20 tokens. A liquidity pool allows users to deposit, withdraw, and swap these assets.

Liquidity Pool Arity: The number of different assets being managed by the liquidity pool.

Liquidity Pool Reserves: A vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, where n is the liquidity pool arity and x_i is the amount of i-th asset currently being held by the liquidity pool.

Liquidity Pool Shells: ERC-20 tokens representing fungible shares in the liquidity pool and the assets being held by it.

Liquidity Pool State: A pair $\langle [x_1, x_2, \ldots, x_n], T \rangle$, where $[x_1, x_2, \ldots, x_n]$ is liquidity pool reserves and T is the total amount of the liquidity pool shells in circulation.

Deposit: An act of putting additional assets into a liquidity pool in exchange for newly created shells.

Withdrawal: An act of taking assets from a liquidity pool in exchange for shells being removed from circulation.

Swap: An act of putting a certain amount of one asset into a liquidity pool in exchange for a certain amount of another asset.

6.2 Behaviors

With these terms defined, we now need to define the key behaviors. Namely, we have deposits, swaps and withdrawals. In a deposit, the user adds tokens to the pool. In return, she receives shells. In a withdrawal, the user withdraws tokens by redeeming her shells. In a swap, the user exchanges one reserve token for another.

We define the **deposit function** as:

$$\mathcal{D}\left(\langle [x_1, x_2, \ldots, x_n], T\rangle, [x'_1, x'_2, \ldots, x'_n]\right) = y \tag{1}$$

Here $\langle [x_1, x_2, \ldots, x_n], T\rangle$ is the liquidity pool state before the deposit, $[x'_1, x'_2, \ldots, x'_n]$ is the vector of asset amounts added to the pool, and y is the amount of shells created and given to the depositor in exchange for the assets. The state after the deposit can be represented as:

$$\langle [x_1 + x'_1, x_2 + x'_2, \ldots, x_n + x'_n], T + y\rangle$$

Similar to the deposit function, we define the **withdrawal function** as:

$$\mathcal{W}\left(\langle [x_1, x_2, \ldots, x_n], T\rangle, [x'_1, x'_2, \ldots, x'_n]\right) = y \tag{2}$$

Here $\langle [x_1, x_2, \ldots, x_n], T\rangle$ is the liquidity pool state before the withdrawal, $[x'_1, x'_2, \ldots, x'_n]$ is the vector of asset amounts withdrawn from the pool, and y is the amount of shells redeemed by the withdrawer in exchange for assets. These shells are permanently removed from circulation. The state after the withdrawal can be represented as:

$$\langle [x_1 - x'_1, x_2 - x'_2, \ldots, x_n - x'_n], T - y\rangle$$

For the **swap function**, we define it as follows. For every $i \in \{1 \ldots n\}$ and $j \in \{1 \ldots n\} \setminus \{i\}$ we define the swap function for exchanging i-th asset to j-th asset like this:

$$\mathcal{S}_{i,j}\left(\langle [x_1, x_2, \ldots, x_n], T\rangle, x\right) = y \tag{3}$$

Here $\langle [x_1, x_2, \ldots, x_n], T\rangle$ is the liquidity pool state before the swap, x is the amount of i asset given to the pool, y is the amount of j-th asset given to the swap initiator. We will refer to x as the swap "origin amount" and i as the "origin token." And y as the swap "target amount" and j as the "target token." This is because the swap initiator begins with x, the origin of the trade, and finishes with y, the target of the trade. The state after the swap can be represented as: $\langle [x'_1, x'_2, \ldots, x'_n], T\rangle$, where:

$$x'_k = \begin{cases} x_k + x, \text{ if } k = i \\ x_k - y, \text{ if } k = j \\ x_k, \text{ otherwise} \end{cases}$$

6.3 Utility function

A liquidity pool must be able to compute the relative values between reserves and shells. How can we create programmable rules that assess value, an inherently subjective concept? The Shell Protocol employs a **utility function**, U. Utility functions assign a value score to elements in a set of alternatives. When an agent (such as a liquidity pool) needs to choose between two alternatives, they pick whichever has the higher utility score. In the case of Shell, the inputs to the utility function are the balances of the token reserve, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, and the output is a scalar value score. The Shell pool's utility function has two components, the **gain function**, G, and the **fee function**, F.

$$U(\mathbf{x}) = G(\mathbf{x}) - F(\mathbf{x})$$

$$G(\mathbf{x}) = \sum x_i$$

$$F(\mathbf{x}) = \sum f_i(\mathbf{x})$$

The utility function is the difference between the gain function and the fee function. The gain function is the sum of all the reserve balances held in the pool. The fee function is the sum of the micro fee functions for each individual token, f_i. Most of the complexity of the pool's utility function, and thus in the pool's logic, is embedded in the micro fee function. At a high level, f_i is zero when the reserve token's actual weight in the pool is at or near its ideal weight. However, once a reserve token's actual weight has deviated past a certain threshold, then f_i begins to increase quadratically. When all reserve tokens are sufficiently close to their ideal weights, then the utility function is just G, the sum of all reserve balances. Hence, while the pool is in a balanced state, it will behave like a constant sum automated market maker, which keeps the sum of balances constant during a swap. Constant sum market makers trade tokens at a 1:1 price by default.

In order to deconstruct the micro fee function, we need to define some new variables. The ideal weight for each reserve in the pool, w_i, has the following constraints:

$$w_i \in (0, 1), \quad \sum w_i = 1$$

Using w_i, we can construct the ideal balance for each token, I_i, and the lower and upper fee thresholds, L_i and V_i respectively:

$$I_i = G \cdot w_i$$

$$L_i = I_i \cdot (1 - \beta)$$
$$V_i = I_i \cdot (1 + \beta)$$

Where:

$$\beta \in (0, 1)$$

The lower fee threshold is β percent below the ideal balance, and the upper fee threshold is β percent above the ideal balance. The "no-fee zone" or "no-slippage zone" is a band with its midpoint at I_i whose distance is $2 \cdot \beta \cdot w_i \cdot G$ units long. When the reserve balance falls within this range, $f_i = 0$. To encapsulate this logic, let us define the fee margin, M_i, and the intermediate fee, m_i.

$$M_i = \max(L_i - x_i, x_i - V_i, 0)$$
$$m_i = \Delta \cdot \frac{M_i}{I_i}$$
$$f_i = M_i \cdot \min(m_i, \gamma)$$
$$\Delta > 0$$

The parameter Δ can be thought of as the rate of increase of the fee function. The higher Δ becomes, the faster f_i will increase as the reserve balance goes deeper into the fee-zone. The parameter γ can be thought of as the maximum allowable intermediate fee. In the current implementation, we set $\gamma = \frac{1}{4}$. Including this constraint ensures $F < G$. We can now do some reductions to find a complete expression for f_i:

$$M_i = \max(L_i - x_i, x_i - V_i, 0) = \max(I_i \cdot (1 - \beta) - x_i, x_i - I_I \cdot (1 + \beta), 0) =$$
$$= I_i \cdot \max\left(1 - \beta - \frac{x_i}{I_i}, \frac{x_i}{I_i} - 1 - \beta, 0\right) =$$
$$= I_i \cdot \max\left(\left(1 - \frac{x_i}{I_i}\right) - \beta, \left(\frac{x_i}{I_i} - 1\right) - \beta, 0\right) =$$
$$= I_i \cdot \max\left(\max\left(1 - \frac{x_i}{I_i}, \frac{x_i}{I_i} - 1\right) - \beta, 0\right) =$$
$$= I_i \cdot \max\left(\left|\frac{x_i}{I_i} - 1\right| - \beta, 0\right)$$

And some more reductions:

$$f_i = M_i \cdot \min(m_i, \gamma) = M_i \cdot \min\left(\Delta \cdot \frac{M_i}{I_i}, \gamma\right) =$$
$$= I_i \cdot \max\left(\left|\frac{x_i}{I_i} - 1\right| - \beta, 0\right) \cdot \min\left(\Delta \cdot \frac{I_i \cdot \max\left(\left|\frac{x_i}{I_i} - 1\right| - \beta, 0\right)}{I_i}, \gamma\right) =$$
$$= I_i \cdot \max\left(\left|\frac{x_i}{I_i} - 1\right| - \beta, 0\right) \cdot \min\left(\Delta \cdot \max\left(\left|\frac{x_i}{I_i} - 1\right| - \beta, 0\right), \gamma\right)$$

Putting these reductions together, we arrive at the complete expression for the micro-fee function:

$$f_i = \begin{cases} 0, & \text{if } \left|\frac{x_i}{I_i} - 1\right| \leqslant \beta \\ \Delta \cdot I_i \cdot \left(\left|\frac{x_i}{I_i} - 1\right| - \beta\right)^2, & \text{if } \beta < \left|\frac{x_i}{I_i} - 1\right| \leqslant \beta + \frac{\gamma}{\Delta} \\ \gamma \cdot I_i \cdot \left(\left|\frac{x_i}{I_i} - 1\right| - \beta\right), & \text{if } \beta + \frac{\gamma}{\Delta} < \left|\frac{x_i}{I_i} - 1\right| \end{cases} \tag{4}$$

6.4 Halt thresholds

Before we formalize deposits, withdrawals and swaps, we must describe a mechanic somewhat unrelated to the utility function: halt thresholds. The liquidity pool will not have any direct awareness of prevailing market prices for tokens. It will only know how many reserve tokens it has and how much a user is trying to deposit, withdraw or swap. Therefore, if a stablecoin reserve permanently loses its peg, the pool will not know that this stablecoin is now worthless. Without auxiliary logic, the pool will continue to accept the worthless token until all other reserves have been drained. To prevent this eventuality, the pool will have a minimum and maximum allocation for each token. We can define thresholds for each token i as:

$$\hat{L}_i = I_i * (1 - \alpha)$$

$$\hat{V}_i = I_i * (1 + \alpha)$$

Where:

$$0 < \beta < \alpha < 1$$

And

$$w_i \cdot (1 + \alpha) < 1$$

$\hat{L}_i$ and $\hat{V}_i$ are the lower and upper halt thresholds respectively. The parameter α is the percent above and below the ideal point allowable for a reserve token balance. The pool will only execute transactions (i.e. deposits, withdrawals and swaps) if the resulting state is such that:

$$\hat{L}_i \leq x_i \leq \hat{V}_i$$

6.5 Deposits and withdrawals

The advantage of defining the pool's logic in terms of a utility function is that it becomes relatively straightforward to construct our three main behaviors: deposits, withdrawals and swaps. This section will focus on deposits and withdrawals. Recall that for deposits, the depositor adds reserve tokens to the pool and receives shells in return. The pool logic must therefore determine a conversion rate between the tokens added and the amount of shells created. For withdrawals, the pool must determine a conversion rate between the shells redeemed by the withdrawer and the amount of tokens removed from the pool.

Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ be initial reserves, and $\mathbf{x}' = [x'_1, x'_2, \ldots, x'_n]$ be the amounts of tokens being deposited or withdrawn. Then the following inequality must hold:

$$\frac{U(\mathbf{x})}{T} \leq \frac{U(\mathbf{x} + \mathbf{x}')}{T + y} \tag{5}$$

Where T is the total supply of shells, and y in this case is the amount of shells added to the supply after the deposit or withdrawal. This inequality requires that the ratio between the pool's utility and its shell supply must always either increase or stay the same. For deposits, new shells will be created and $y > 0$. For withdrawals, shells will be removed from circulation and $y < 0$. Equation 5 is the **invariant** of the pool. That is, for every valid transaction, this relationship must remain true. What this inequality means is that if the pool's utility increases by 10%, then the shell supply should increase by no more than 10%. If the pool's utility decreases by 10%, then the shell supply should decrease by at least 10%. Adhering to this constraint, we can construct the deposit and withdrawal functions:

$$\mathcal{D}(\langle \mathbf{x}, T \rangle, \mathbf{x}') = \frac{T}{G(\mathbf{x}) - F(\mathbf{x})} \left(G(\mathbf{x}') - \rho \left(F(\mathbf{x} + \mathbf{x}') - F(\mathbf{x}) \right) \right) \tag{6}$$

$$\mathcal{W}(\langle \mathbf{x}, T \rangle, \mathbf{x}') = \frac{T}{G(\mathbf{x}) - F(\mathbf{x})} \left(G(\mathbf{x}') + \rho \left(F(\mathbf{x} - \mathbf{x}') - F(\mathbf{x}) \right) \right) \tag{7}$$

where

$$\rho\left(x\right) = \begin{cases} x, & \text{if } x > 0 \\ \lambda x, & \text{otherwise} \end{cases}$$

and

$$\lambda \in [0, 1]$$

Ignoring ρ, Equations 6 and 7 are the result of combining Equations 1 and 2 with the invariant from Equation 5. Where does ρ fit in? You can think of it as a **dynamic fee**. When a transaction unbalances the pool and the fee function, F, increases as a result of a transaction, then the shells issued (or redeemed) will simply be proportional to the increase in utility. I.e., $\rho(x) = x$. However, when a transaction rebalances the pool and F decreases, then the change in the shell supply will be offset by λ. Therefore, the penalty from unbalancing a pool will be less than the reward for rebalancing a pool. This difference will be λ percent. So if $\lambda = 0.5$, then the dynamic fee will be half of the price slippage. If $\lambda = 1$, then there will be no dynamic fee. We discuss the implications and mechanics of the dynamic fee in Section 3.4.

6.6 Swaps

Mathematically, a swap is a simultaneous deposit and withdrawal where the pool's utility remains unchanged (see Figure **??**). Intuitively, this should make sense because during a swap, the shell supply does not change, and therefore neither should the pool's utility. To derive the swap

formula, first realize that depositing a negative amount is the same as withdrawing a positive amount:

$$\mathcal{D}(\langle \mathbf{x}, T\rangle, -\mathbf{x}') = \frac{T}{G(\mathbf{x}) - F(\mathbf{x})} \left(G(-\mathbf{x}') - \rho \left(F(\mathbf{x} - \mathbf{x}') - F(\mathbf{x})\right)\right) =$$

$$= -\frac{T}{G(\mathbf{x}) - F(\mathbf{x})} \left(G(\mathbf{x}') + \rho \left(F(\mathbf{x} + \mathbf{x}') - F(\mathbf{x})\right)\right) = \mathcal{W}(\langle \mathbf{x}, T\rangle, \mathbf{x}')$$

Where:

$$\mathbf{x}' = [0, \ldots, 0, x_i' = x, 0, \ldots, 0, x_j' = -y, 0, \ldots, 0]$$

Therefore, a swap operation that does not modify the shell supply must be equivalent to the deposit operation that results in zero shells issued:

$$\mathcal{D}(\langle \mathbf{x}, T\rangle, \mathbf{x}') = 0$$

$$G(\mathbf{x}') - \rho \left(F(\mathbf{x} + \mathbf{x}') - F(\mathbf{x})\right) = 0$$

$$y = x - \rho \left(F(\mathbf{x} + \mathbf{x}') - F(\mathbf{x})\right)$$

Where y is the target amount and x is the origin amount. Combining the above with Equation 3, we arrive at the following formula for swaps:

$$\mathcal{S}_{i,j}\left(\mathbf{x}, T\rangle, x\right) = x - \rho \left(F(\mathbf{x} + \mathbf{x}') - F(\mathbf{x})\right) \tag{8}$$

Although Equation 8 is simple to state, it is difficult to solve analytically because F is a function of y. Therefore, the protocol relies on an iterative approach to finding the correct value for the swap. We can restate Equation 8 as a function, h, of y:

$$\mathcal{S}_{i,j}\left(\mathbf{x}, T\rangle, x\right) = h(y)$$

We then iterate as follows:

$$y_0 = x$$

$$y_k = h(y_{k-1})$$

If the algorithm converges, i.e. $y_k = y_{k-1}$, then we will have found the correct value.

7 Future direction

From a technical standpoint, the end goal for Shell is to create a layer of abstraction above complex stablecoin markets. With such a protocol, end users and developers will feel like they are just using money, similar to how we don't think about the plumbing when we turn on the tap water. To support the diversity of use cases and market conditions, we need a generalized liquidity pool model. Consider the rise of Ethereum. Bitcoin is a blockchain built for a specific purpose: tracking who owns how many bitcoins. Ethereum, in contrast, is a general purpose

blockchain that acts more as an operating system. The general nature of Ethereum is why DeFi is built on Ethereum instead of Bitcoin, despite the latter being more established and more valuable. When Ethereum launched in 2015, no one knew what DeFi would look like (the term hadn't even been invented yet). Ethereum gave developers a platform for experimentation. In turn, developers discovered new, unexpected use cases, which allowed the Ethereum ecosystem to grow organically.

7.1 Liquidity pool "operating system"

A fundamental design goal for Shell is to build an operating system for liquidity pools. Rather than coming up with innovative AMM strategies ourselves, it is better to create a platform for others to implement their own creative ideas. This is analogous to how Ethereum created a platform for innovative DeFi applications. There are many facets to a liquidity pool, but the most important is the utility function (see Section 6.3). Upcoming iterations of Shell will offer increasingly flexible utility functions to truly customize the behavior of the pool.

7.2 Governance

The second design goal is a governance layer for the protocol itself. An in-depth discussion of Shell Protocol governance is beyond the scope of this paper, which focuses on the core protocol. Suffice it to say that Shell will be governed by a community of token holders. This token will also play a role in incentivizing growth of the protocol. It should also be mentioned that pools are intended to be controlled and managed not by the core protocol but by whomever deploys and subsequently operates the pools. The core protocol's role in governance will be to coordinate action beyond the purview of a single pool, such as protocol-wide growth incentives.

7.3 Pool aggregation

The third design goal for the Shell Protocol is to create an aggregation layer (on-chain) to link pools together. It will allow any shell to convert into any other shell. It will also combine the liquidity of all the pools into a liquidity ocean. The true value of the Shell Protocol will be in this aggregation layer. Developing flexible pool models is an important step and a challenge in its own right. But the ultimate destination is the liquidity ocean.

8 Conclusion

The model presented in this paper is the first of many iterations. It has weights, deep liquidity, protections against a broken peg and dynamic fees. The behavior of the model can be adjusted dynamically by changing parameters. This allows pools to adapt to new use cases and market conditions. Shell will ultimately create a liquidity pool model that can act as an operating system for stablecoin AMMs. Given the superiority of shell tokens versus individual stablecoins, shells will eventually become the primary means of storing and transacting value. The mission for

the Shell Protocol is to create an internet monetary system. To realize this vision, we need an aggregation layer that will combine liquidity from the myriad pools. Shell's governance will not exist to manage individual pools, but to oversee the liquidity ocean.

References

[1] Aave (2020, September 4). "Aave FAQs." https://docs.aave.com/faq/

[2] Adams, Hayden (2019, July 5). "Uniswap white paper". https://hackmd.io/C-DvwDSfSxuh-Gd4WKE_ig#DEX-inside-a-Whitepaper

[3] Binance Academy (2020, September 4). "What Is Yield Farming in Decentralized Finance (DeFi)?" https://academy.binance.com/blockchain/what-is-yield-farming-in-decentralized-finance-defi

[4] Buterin, Vitalik (2018, March). "Improving front running resistance of x*y=k market makers". *ETH Research*. Retrieved 2019, November 24: https://ethresear.ch/t/improving-front-running-resistance-of-x-y-k-market-makers/1281

[5] Circle (2020). "USDC: the fastest growing, fully reserved digital dollar stablecoin." Retrieved on 2020, September 27. https://www.circle.com/en/usdc

[6] Cohen, Michael (2020, April). "The Rise of sUSD." https://synthetix.community/blog/2020/04/13/the-rise-of-susd

[7] Coin Market Cap (Aug, 2020). https://coinmarketcap.com/

[8] DeFi Pulse (2020, September 4). "DeFi Pulse: Compound". Retrieved 2020, September 4. https://defipulse.com/compound

[9] DeFi Pulse (2020, September 27). "DeFi Pulse: Curve Finance". Retrieved 2020, September 27. https://defipulse.com/curve-finance

[10] De la Rouviere, Simon (2017, Nov 21). "Tokens 2.0: Curved Token Bonding in Curation Markets". Retrieved, 2020, September 27. https://medium.com/@simondlr/tokens-2-0-curved-token-bonding-in-curation-markets-1764a2e0bee5

[11] De, Nikhilesh (September 21, 2020). "SEC, OCC Issue First Regulatory Clarifications for Stablecoins." https://www.coindesk.com/occ-banks-can-hold-some-stablecoin-reserves

[12] Dogen (1240). *The Mountain and Waters Sutra* (山水經). Translated by Arnold Kotler and Kazuaki Tanahashi (2008). Hackett Publishing Company.

[13] Egorov, Michael (2019, November 10). "StableSwap - efficient mechanism for Stablecoin liquidity." https://www.curve.fi/stableswap-paper.pdf

[14] Fedwire (2020). "Fedwire Funds Service - Monthly Statistics." https://www.frbservices.org/resources/financial-services/wires/volume-value-stats/monthly-stats.html

[15] Huthinson, Emma (2016). *Principles of Microeconomics*. Chapter 6.2: The Indifference Curve. University of Victoria. https://pressbooks.bccampus.ca/uvicecon103/chapter/6-3-how-changes-in-income-and-prices-affect-consumption-choices/

[16] Kereiakes et al (2019, April). "Terra Money: Stability and Adoption". https://s3.ap-northeast-2.amazonaws.com/terra.money.home/static/Terra_White_paper.pdf?201904

[17] Leshner, Robert and Geoffrey Hayes (2019, February). "Compound: The Money Market Protocol". https://compound.finance/documents/Compound.Whitepaper.pdf

[18] Maker DAO (2020). "The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System." https://makerdao.com/en/whitepaper/#abstract

[19] Martinelli, Fernando and Nikolai Mushegian (2019, September 19). "Balancer: A non-custodial portfolio manager, liquidity provider, and price sensor." https://balancer.finance/whitepaper.html

[20] mStable (2020). Accessed on 2020, September 29. https://mstable.org/

[21] Nakamoto, Satoshi (August, 2008). "Bitcoin: A Peer-to-Peer Electronic Cash System". https://bitcoin.org/bitcoin.pdf

[22] Opyn (September 2020). "Securing Decentralized Finance." https://opyn.co/

[23] Paxos (2020). "The New Digital Dollar." Retrived on 2020, September 27. https://www.paxos.com/pax/

[24] Peter, Bianca (2019, September). "Bank Market Share by Deposits and Assets." https://wallethub.com/edu/sa/bank-market-share-by-deposits/25587/

[25] Swerve Finance (2020). Retrieved on 2020, September 27. https://swerve.fi/

[26] Szabo, Nick (2017, February). "Money, blockchains, and social scalability." http://unenumerated.blogspot.com/2017/02/money-blockchains-and-social-scalability.html

[27] Tether (2016, June). "Tether White Paper". https://tether.to/wp-content/uploads/2016/06/TetherWhitePaper.pdf

[28] The US Department of State: Office of the Historian (2018). "Nixon and the End of the Bretton Woods System, 1971–1973". Retrieved 2019, July 21: https://history.state.gov/milestones/1969-1976/nixon-shock

[29] yEarn home page (2020, September 4). https://yearn.finance/

[30] Zapper home page (2020, September 30). "Manage your DeFi assets and liabilities in one simple interface.". https://zapper.fi/