Single-Sided Automated Market Maker for European Options

contact@pods.finance

August 2021

Abstract

We present a single-sided automated market maker (AMM) approach for European options on EVM (Ethereum Virtual Machine) compatible blockchains. The solution optimizes for liquidity provider (LP) value preservation over time and algorithmically updates the options price. Virtual balances keep track of the pool's debts and assets to further distribute the proceeds fairly among LPs based on their initial contribution. Our approach uses implied volatility as an initial input when creating a liquidity pool and, as trading takes place, the AMM calculates the next implied volatility with a finite difference method and solves for the updated option price via Black-Scholes.

1 Introduction

An option is a financial instrument that gives the buyer the right to sell or buy an underlying asset in the future at a particular strike price. Although the buyer has the right, he does not have the obligation to exercise the option. The seller of the contract, on the other hand, is obligated to sell or buy upon exercise at the expiry date. In order to compensate for the risk of being obligated to sell or buy at a particular price, the option seller receives a premium that should represent the payoff of the option in the future weighted by the probability of exercise [1].

Traditionally, options are traded by investment banks, hedge funds, and large companies on listed exchanges and on over-the-counter desks. They are used mostly to fulfill investment strategies, hedge assets, and for market making activities. As of today, most of the market players are used to trading options on orderbooks. With the surge of general purpose Automated Market Makers (AMMs) such as Uniswap [2], Balancer [3], and Curve [4], trading options became possible for DeFi users as well. However, impermanent loss [5], specifically in the general purpose pools used for options tokens became an issue of quantifiable concern for liquidity providers (LPs). With advancement in Uniswap's v3 protocol to concentrate liquidity in a price range [6], a strategy that mimics

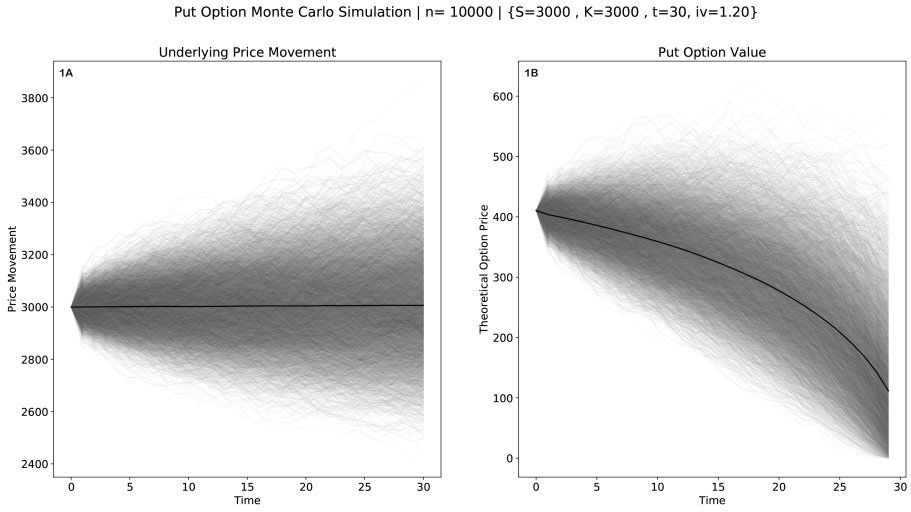

Figure 1: Time decay of Black-Scholes put option price with a starting underlying price of 3000 and a strike price of 3000 expiring in 30 days. The average value of the put option, outlined in black in 1B, declines in comparison to the average value of the underlying in 1A.

the payoff of being short volatility, the possibility to hedge such an LP position with vanilla options became evident [7].

The price discovery mechanism of general-purpose AMMs normally relies exclusively on the volume of trades, assuming that in a liquid market the assets should be priced correctly by market forces. However, DeFi options markets are still nascent by the time of this paper and options may not be updated as often, leaving the price outdated, potentially generating a substantial impermanent loss for the LPs of the option pool.

In addition to the lack of frequent trading activity, most of the general-purpose AMMs face the problem of options prices losing value due to the passage of time (Figure 1), meaning arbitrageurs would face difficulty in profiting from such an AMM. The constant product approach of $x * y = k$ will struggle to reflect an option price as an option decays with time despite a balance of each token staying the same. One can even have a case of an option decaying to the value of zero over time, which can blow up the constant product formula. In this paper we propose a modified constant-product invariant approach that bases the effective x and y values on the Black-Scholes pricing model. This AMM changes the option price by updating a parameter called implied volatility in addition to the pool constant k whenever a trade takes place.

2 Pods Options

We introduce an AMM for European, physically settled, fully collateralized call and put options. It allows for the creation of options tokens outside of the liquidity pool that can be supplied to the pool as liquidity paired with a stablecoin position. Options can also be further fractionalized, making it unnecessary to collateralize one whole ETH to mint an option.

The options are organized in series. Each option series represents a certain combination of characteristics such as: expiration date, strike price, and asset pair (e.g ETH:DAI, ETH:wBTC, ETH:aUSDC). Additionally, each option series has its own ERC20 equivalent token and in the first implementation of this model, its own AMM pool. The collateral provided can be placed as an interest-bearing token (aTokens) from Aave [8], meaning users can increase the productivity of their assets and gain additional yield while providing liquidity to the AMM or selling options.

Selling fully collateralized options and placing yield-bearing tokens as collateral creates a new type of product for DeFi users and can be viewed as a synthetic structured product from a TradFi perspective. By tokenizing the seller's position, the protocol allows users to create further structures such as advanced options strategies (vertical spreads, calendar spreads, married puts) and also arbitrage strategies via Put-Call Parity (sell a put ATM and buy a call ATM with an expiration date matching a futures contract of the underlying).

3 Options Pricing

The price of an option depends on the behavior of implied volatility. An increase in the volatility of the underlying price σ_{iv} can increase the option value because the more the underlying price fluctuates, the higher the probability that the option may expire ITM by crossing the strike price. This volatility is expressed as σ_{iv} and is the only parameter needed as an initial input for option pricing on our AMM as it is used to discover the initial price with the provided implied volatility term σ_{iv}.

When an options pool is created by an LP, the LP has to define the price of the option by setting an initial implied volatility at the moment of creation. One could think of the σ_{iv} as a factor that is guessed by the first LP. If the LP sets σ_{iv} too low relative to the σ_{iv} on centralized exchanges, then the options become underpriced and buying will drive σ_{iv} up. If the LP sets σ_{iv} too high, it can encourage people to sell options into the pool and drive σ_{iv} down.

To price options, we use an implementation of the Black-Scholes formula created by Black and Scholes [9] outlined below for puts P with the risk-free rate r set to zero with S representing the value of the underlying, K being the strike price, and σ_{iv} as implied volatility:

$$\mathrm{P(S,t)} = \mathrm{e}^{-rt}\mathrm{K}N(-\mathrm{d}_2) - \mathrm{S}N(-\mathrm{d}_1)$$

$$\mathrm{d}_1 = \left[\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma_{iv}^2}{2}\right)t\right]\frac{1}{\sigma_{iv}\sqrt{\mathrm{t}}}$$

$$\mathrm{d}_2 = \left[\ln\left(\frac{S}{K}\right) + \left(r - \frac{\sigma_{iv}^2}{2}\right)t\right]\frac{1}{\sigma_{iv}\sqrt{\mathrm{t}}}$$

$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} \mathrm{e}^{-\frac{z^2}{2}}dz$$

The dynamics of the above equation are visible in Figure 1B. Upon minting an option, the further out the expiration date is, the more time there is for the underlying price to cross the strike price, but as the time to maturity approaches, this time begins to dwindle and with it the chance of the option expiring in the money (ITM), resulting in the price of the option dropping. This phenomenon is known as theta decay. Additionally, the further away the underlying price moves from the strike, the less likely the strike intersection and the less valuable the option becomes.

Updating Implied Volatility In order to correctly price the option one has to update the implied volatility parameter. Failure to do so can lead to option price manipulation and mispricing of the option. To price the option, after the initial σ_{iv} has been set, a trade at any given instant i can be initiated. After the trade, the pool's inventory of option tokens shifts, changing with it the pool constant, and the AMM presents a new price. This price will be used to guess the next implied volatility.

The protocol uses a finite difference method presented by Yan Wang in the paper "A Well-Posed Algorithm to Recover Implied Volatility" [10] to guess the implied volatility correspondent to the new equilibrium price in a gas-optimal manner.

After the protocol finds the implied volatility that translates the new pool state it will store this value for the next trade. To avoid price manipulation the protocol contains an implied volatility oracle that can add boundaries to how far the new implied volatility can diverge, stemming from the logic that volatility cannot go to infinity or to zero as time evolves. The oracle factor is modular and the LP can decide how effective it will be in comparison to the calculated implied volatility. As the next trade unfolds, the newly calculated implied volatility will be used in conjunction with the implied volatility oracle to calculate the next price with the entire process repeating as visualized in Figure 2 below.

Additionally, the protocol contains dynamic fees for the AMM in addition to a fixed fee. The purpose of dynamic fees is to discourage economic attacks observed in traditional options markets [11] in the form of the manipulation of implied volatility through rapid and increasing buying/selling or flash loan attacks. If attacks are attempted, then the increasing fees go to the LPs. The calculation of the dynamic fee is dependent on the size of the trade relative to the pool amount A:

$$DynamicFee = \frac{\alpha * tradeAmount_A^3}{poolAmount_A^3}/100$$

Where parameter $\alpha = 2000$. The dynamic fees begin to take effect when large chunks of options are being traded and are negligible for small trades. We also aim to balance fees in such a way as to satisfy the LPs for the extra risk they may face if a large buyer/seller attempts to manipulate the options price.

4 AMM General Logic

The AMM functions on a debt-to-asset logic, meaning that assets are deposited by an LP to the AMM with the AMM owing a debt to be returned back at a future point in the form of either option tokens or stablecoins. Unlike a Uniswap approach of LP position tokenization, there is no token generated upon staking with Pods at this point. The AMM acts as a single-sided pool since it accepts user's deposits in whatever proportion the user wants to provide and it will track this position until the end of the option expiration.

During each interaction, the AMM calculates how much it owes to LPs and how much it has in assets. When an LP adds liquidity to the pool (regardless of the amount of options tokens or stablecoins added) their balance is updated to reflect the pool's total debt.

In case the LP elects to remove funds from the pool the AMM will give back the LPs provision with a combination of the both assets (options tokens and stablecoins) to keep the LP's initial deposit's exposure.

There are two main sources of returns for a liquidity provider: the AMM returns and the fees. Factors like the historical option premium price, the initial implied volatility, and the trading volume of options in the lifetime of the option impact the result obtained by LPs. The LPs are exposed to the AMM returns for only the time and the representativeness of their liquidity.

The AMM will return its debt to LPs by the time they decide to remove liquidity by distributing the assets to offset their initial debt value by using a combination of the assets the AMM currently has. The fees are added to the user's position at the time of removal.

Pricing an option upon trading activity triggers two mechanisms: updating the theoretical Black-Scholes price and the modified constant-product invariant. The pricing and LP positions are disconnected modules, making it possible to use the AMM for other assets when using a different pricing mechanism.

Below we outline the logic of the AMM considering a market price P_i.

5 Mathematical Formulation of AMM

The AMM has to factor into account the dynamics of option decay to keep track of the correct LP position. The AMM logic considers that every pool consists of a balance of options tokens (fungible ERC20s from the same series) and

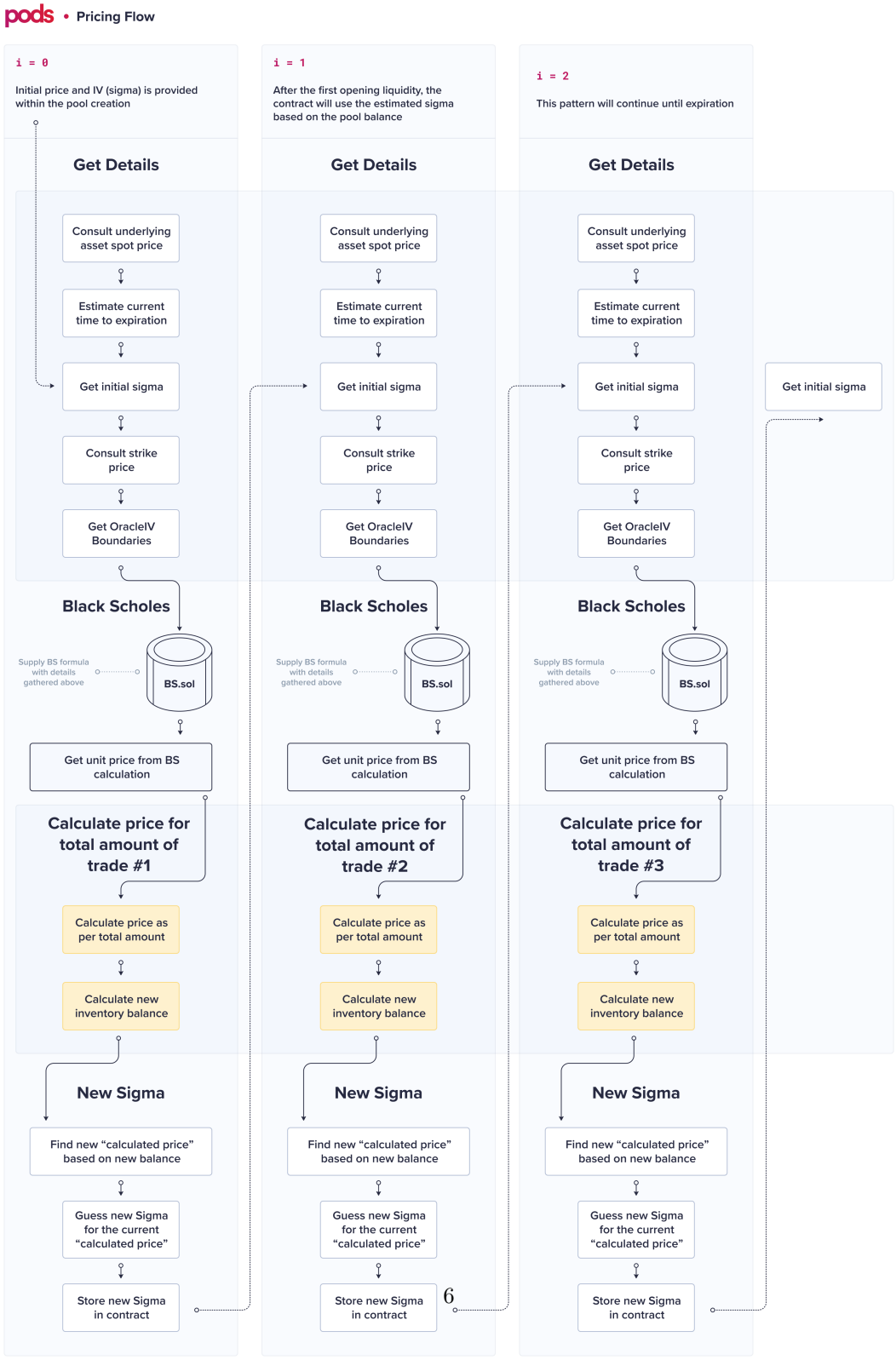

Figure 2: The Cycle of Updating The Option Price.

stablecoins (DAI, USDT, USDC, or aUSDC). The options tokens can be bought or sold to the AMM pool at any time as the AMM algorithmically calculates the updated premium, taking into account the factors of the movement in the underlying price S and the implied volatility σ_{iv}. The following events E_i can happen in the AMM in a given instant i: trade T, add liquidity AL, or remove liquidity RL.

$$E_i \in \{T, AL, RL\}$$

Derivative formulations contain one or more data points from the market, e.g. spot price S_i from Chainlink, and one or more "free parameters", which will be called, hereinafter, Degrees of Freedom or DoF, e.g. implied volatility σ_{iv} in options or interest rate r in futures. The theoretical price P_i of an option (put or call) is calculated by the prediction from the Black-Scholes implied volatility algorithm rather than the liquidity of the pool in the AMM.

Trading The opening option price (theoretical option price) P_i, i.e., price when an event takes place, is defined as a function f_p using as inputs an external vector $MarketData$ and internal current DoF. The price will be defined as the quantity of token B (stablecoins) needed to buy one token A (options token) in a given time.

$$P_i = f_p(DoF_{i-1}, MarketData_i)$$

The trading event triggers a transfer of amounts of token A and token B using, without any loss of generality, a function relating the transferred amount of A defined as A_i and a transferred amount B as B_i. The function can be defined for each event using a certain rule where the new DoF will be calculated to guarantee:

$$f_p(DoF_{i-1}, MarketData_i) = -\frac{B_i}{A_i}$$

In the current implementation, the rule chosen to keep this property is a modified product constant formulation where the boundaries are set on the minimum amount of tokens each of the sides has, represented as virtual pool amounts, given the last option price. One can think of these minimums as circuit breakers that allow LPs to recover their funds incase of rapid changes in the option price. This linear optimization approach of the minimums is necessary in order to track the price of the option. TB_A and TB_B represent the original invested option and stablecoin balances handed over to the AMM by the LP.

$$poolAmountA = min(TB_A, \frac{TB_B}{P_i})$$

$$poolAmountB = min(TB_B, TB_A * P_i)$$

The modified product constant is updated with one of the two minimums being triggered:

$$k = poolAmountA * poolAmountB$$

The transaction cost is derived by:

$$B_i = \frac{k}{poolAmountA - tradeAmountA} - poolAmountB + Fees$$

And the new unit price after a trade takes place is:

$$TargetPrice_i = \frac{poolAmountA - B_i}{poolAmountA + A_i}$$

The target price is the new equilibrium price after the trade and from there we will be able to calculate Black-Scholes backwards and estimate the new implied volatility σ_{iv} - an internal factor of the model.

Add Liquidity In order to keep our AMM modular in its architecture to allow for advanced structural products in the future, we avoided the LP tokenization approach upon liquidity provision or the representation of LP positions as ERC721s. Our outlined approach involves a debt-to-asset tracking metric known as the pool value factor F_{v_i} instead of an LP token commonly received upon liquidity provision as seen on Uniswap.

To be more precise, when an LP adds liquidity to the Pods AMM in the form of options (token A) or stablecoins (token B), unlike Uniswap tokenization, the add liquidity event triggers the calculation below for the user u balance UB_u, with original deposit of A_{du} and B_{du}, with du representing the time event, the pool total balance TB_i, the pool debt balance DB_i representing the original invested balances handed over to the AMM by the LP, and the **impermanent gain/loss** F_{v_i} also known as the **pool value factor**.

This pool value factor aims to express the current pool circumstances in one number as if it was a snapshot. For example, when an LP added liquidity to the pool, suppose the F_v at that moment was 0.9. This number will be stored in the UB_{F_v} variable within the user's struct. By the time the user wishes to remove the funds, the UB_{F_v} will be compared to the current F_v of the pool, let's say now the factor is at 1. That will represent that the user had an impermanent gain of 11 percent (an example of how F_v evolves towards an impermanent gain during live trading can be seen in section 7).

The calculation for F_v is as follows:

If $i = 0$, then $F_{v_i} = 1$, else

$$F_{v_i} = \frac{TB_{A_{i-1}} * P_i + TB_{B_{i-1}}}{DB_{A_{i-1}} * P_i + DB_{B_{i-1}}}$$

User Balances are virtual balances of the user's liquidity while to the pool they represent a debt the pool has with the LPs.

$$UB_{A_u} = A_{du}$$

$$UB_{B_u} = B_{du}$$

$UB_{F_u} = F_{v_{du}}$, with du represent the liquidity added at a particular time event.

The pools' total assets are represented in the TB variables.

$$TB_{A_i} = TB_{A_{i-1}} + A_{du}$$

$$TB_{B_i} = TB_{B_{i-1}} + B_{du}$$

The pools' total open debt is represented in the debt below. Whenever a user removes liquidity from the pool the debt balance reduces accordingly.

$$DB_{A_i} = DB_{A_{i-1}} + \frac{A_{du}}{F_{v_i}}$$

$$DB_{B_i} = DB_{B_{i-1}} + \frac{B_{du}}{F_{v_i}}$$

Re-add Liquidity If the user has previously added liquidity the user balance UB will be updated:

$$UB_{A_u} = UB_{A_{u-1}} + \frac{F_{v_{du}}}{F_{v_{du-1}}} + A_{du}$$

$$UB_{B_u} = UB_{B_{u-1}} + \frac{F_{v_{du}}}{F_{v_{du-1}}} + B_{du}$$

$$UB_{F_u} = F_{v_{du}}$$

Remove Liquidity The following calculations will be triggered upon removing liquidity. Without loss of generality, the user can withdraw a percentage of the original deposit w_A for token A and w_B for token B:

$$w_A \leq 1 \text{ and } w_B \leq 1$$

Calculate latest price:

$$P_i = f_p(DoF_{i-1}, MarketData_i)$$

If $i = 0$, then $F_{v_{i=0}} = 1$, else calculate F_v:

$$F_{v_i} = \frac{TB_{A_{i-1}} \cdot P_i + TB_{B_{i-1}}}{DB_{A_{i-1}} \cdot P_i + DB_{B_{i-1}}}$$

Let mAA be the amount of A a user can withdraw for each A that the user originally deposited. Let mBB be the amount of B a user can withdraw for each B that the user originally deposited. Let mAB be the amount of B a user

can withdraw for each A that the user originally deposited. Let mBA be the amount of A a user can withdraw for each B that the user originally deposited, then:

$$mAA_i = \frac{min(F_{v_i} \cdot DB_{A_{i-1}}, TB_{A_{i-1}})}{DB_{A_{i-1}}}; mAA_0 = 1$$

$$mBB_i = \frac{min(F_{v_i} \cdot DB_{B_{i-1}}, TB_{B_{i-1}})}{DB_{B_{i-1}}}; mBB_0 = 1$$

$$mAB_i = \frac{TB_{B_{i-1}} - mBB_i \cdot DB_{B_{i-1}}}{DB_{A_{i-1}}}; mAB_0 = 0$$

$$mBA_i = \frac{TB_{A_{i-1}} - mAA_i \cdot DB_{A_{i-1}}}{DB_{B_{i-1}}}; mBA_0 = 0$$

The AMM calculation for updating the user balance after withdrawal is presented in Appendix 9.3.

6 Scenarios, Simulations, and Experiments

Consider a scenario of an ATM put option with a strike of 3000 expiring in 50 days. An LP that provides liquidity of 100 put options TB_{A_i} and the equivalent value of stablecoins TB_{B_i} may face a scenario where more option selling on the AMM drives the total balance of options TB_{A_i} up by 10% as shown in Figure 3. The imbalance in total balances shifts the impermanent gain/loss F_v and the extrinsic value of the option drops as the expiration date approaches. Note that trading fees, being a function of option price, can increase if an option moves further in-the-money, thereby cushioning the risk of exercise for LPs. Alternatively, an LP may face a scenario where there are more option buyers for his options as the option's extrinsic value decays with time, resulting in an impermanent gain as shown in Figure 4.

In the outlined charts below the top left graph represents the movement of the underlying. For the sake of simplicity we set it to equal the strike price. The bottom left is the value of the put option as it drops in value due to theta decay. The top middle graph is the total option pool with the bottom middle graph is the stablecoin pool. The top right graph represents the impermanent gain or loss incurred by the LP with the bottom right being the averaged-out 3% fees generated through trading.

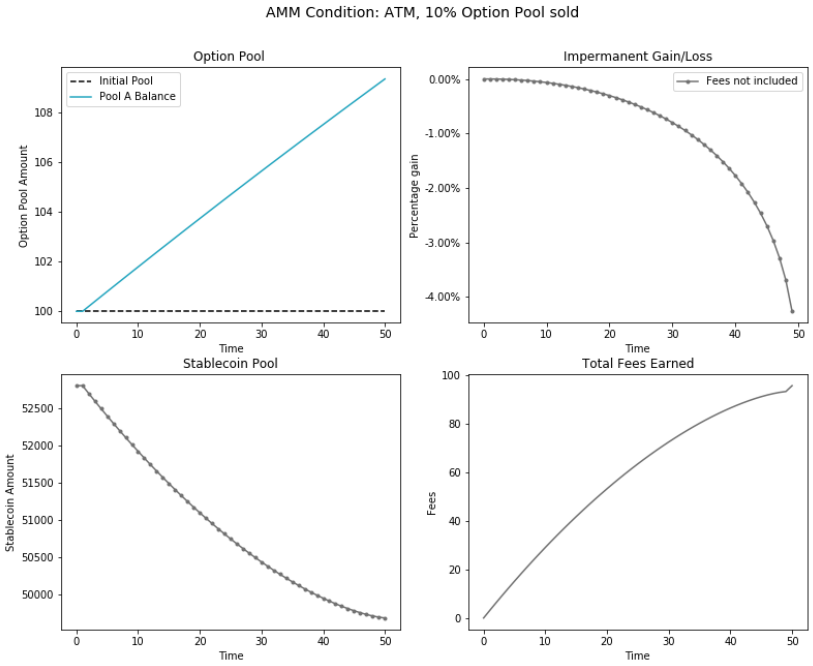

Figure 3: More option selling after LP provides liquidity visible in top left figure.

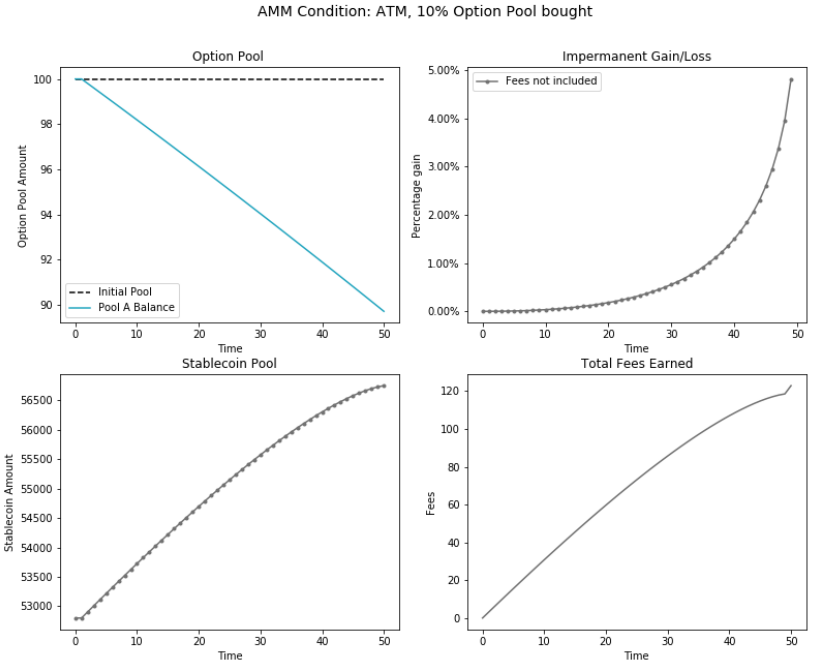

Figure 4: More option buying after LP provides liquidity visible in top left figure.

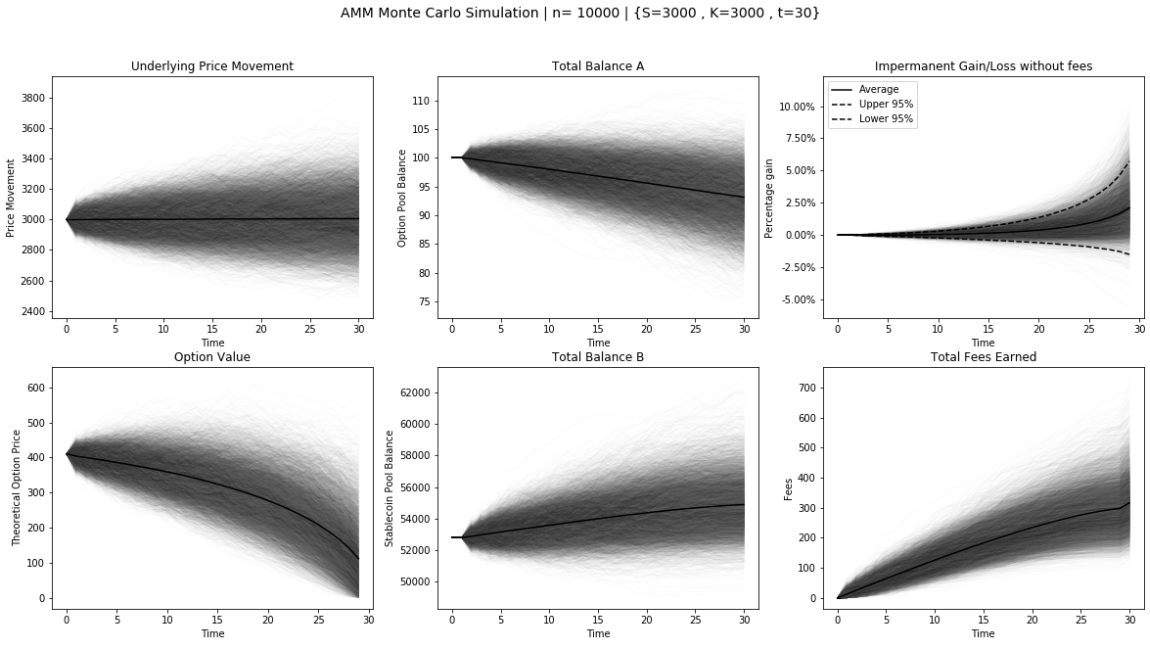

Figure 5: We stochastically simulate 10,000 paths of the underlying asset S at a price of 3000 and a put option with a strike of 3000 expiring in 30 days with a drift rate of 0, resulting in OTM, ATM, and ITM scenarios in the first column. Afterwards, we stochastically simulate trading behavior of agents buying and selling on the AMM, resulting in imbalances between tokens A and B in the middle column with a drift rate of -0.1 (meaning that, on average, there were 10% more option buyers than sellers over 30 days). In the third column we show an LP's expected impermanent gain/loss within 95% confidence intervals with fees outlined separately, demonstrating avoidance of significant impermanent loss, on average, with our AMM.

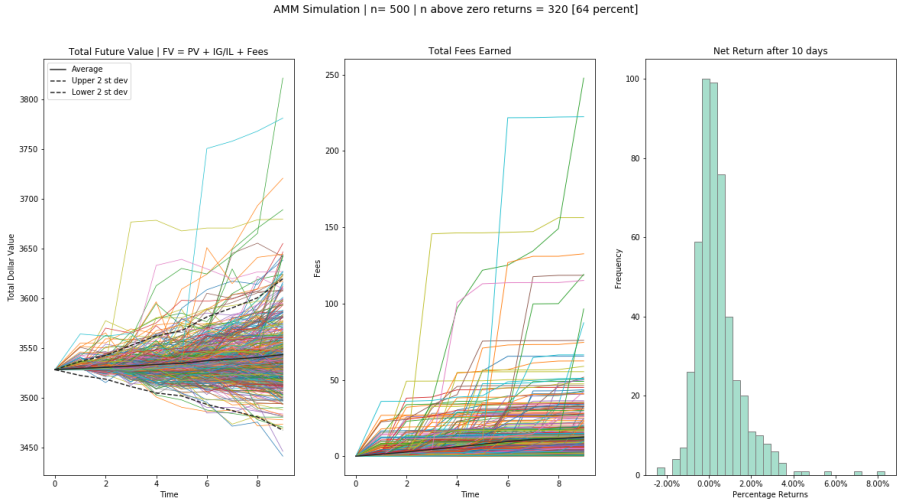

Figure 6: MATIC simulation of potential outcomes for LPs over ten days, assuming 300 MATIC options being traded.

MATIC Experiment During our guarded launch we conducted live experiments on the Polygon network to see how our AMM behaves under market conditions operating under the assumption that 300 MATIC options would be traded over the course of 10 days. The initial put option was valued at 0.22 by the initial LP with the underlying MATIC token price being 2.00 and the strike price being 1.70 with implied volatility at 275% . The pool began with 1000 put option tokens and 1764 in USDC stablecoins. We used The Graph to observe the AMM performance of real trading activity.

Observations The price of MATIC over the ten days (440 events) started at 2.00 and declined towards 1.40, crossing the 1.70 strike and expiring ITM. The option price increased from 0.22 to 0.30 by the expiration date. Implied volatility spiked in the beginning of trading and mean-reverted as is expected. An attempt was made to rapidly buy and sell a chunk of options in the early days at Event 50, as observed in Figures 7, but this resulted in extra fees being generated for LPs. We can observe a slight impermanent gain of 2 percent in Figure 8 with increasing volatility as the option begins to enter ITM.In this case the AMM demonstrated its effectiveness in limiting impermanent loss as well as generating fees from rapid shocks.

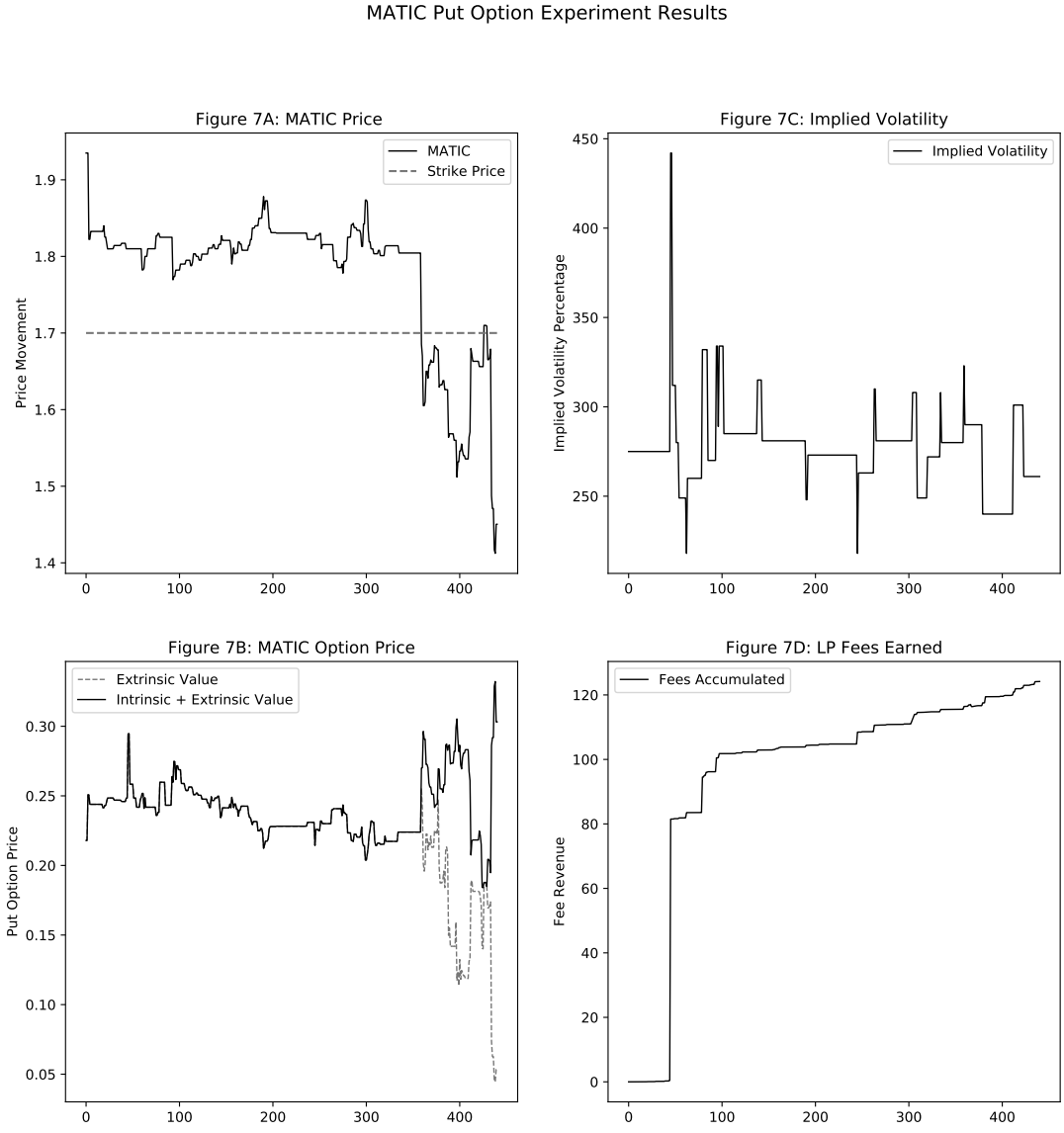

Figure 7: Price movement of the underlying began to increase as expiration approached in 7A. Note the spike in implied volatility and option price in the early days in 7B and 7C. The large fee spike in 7D at event 50 was responsible for over half the revenue due to dynamic fees, which were triggered because of either a potential economic attack, due to the pattern of large buying and then instant selling spiking volatility, or a large/careless block order at Event 50.

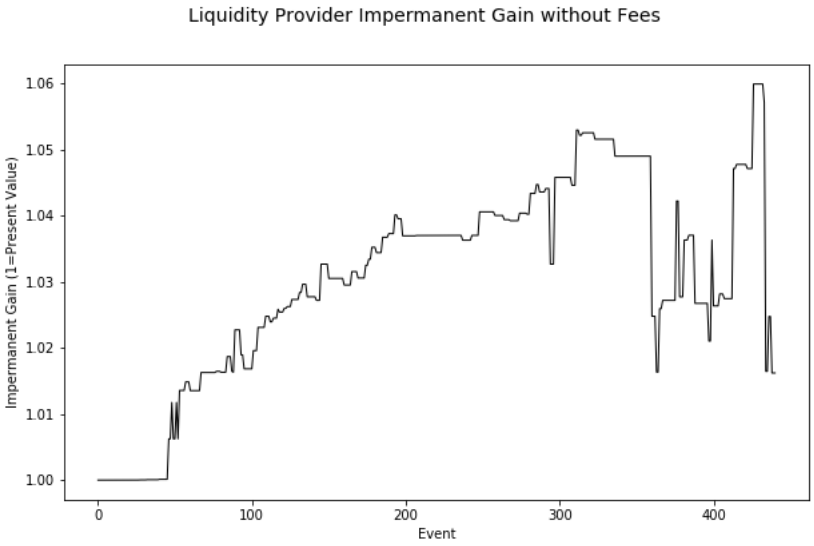

Figure 8: Impermanent gain was achieved in this case with the pool value factor increasing by 2% by the exercise date, which falls within our predicted range.

7 Future Work

This is the first public AMM model we have released with the main limitations of the current model being capital efficiency. Physically settled options require large amounts of funds to be locked in the protocol for a trade to happen, making it less capital efficient.

One improvement moving forward is allowing multiple options series in the same pool. This way the stablecoin liquidity would be shared among all the options tokens available for an asset.

Additionally, instead of building our own liquidation system to allow for partial collateralization, it may make sense to improve the capital efficiency of the options available for trading with the use of defi legos such as Gearbox [12].

Moving forward we aim to improve these gaps in the model and improve the user's experience by developing an automatic re-add liquidity function for rolling the liquidity from one expired option to the next by evolving the liquidity provision action to a non-expiring derivative instrument.

Acknowledgements

This paper was built and reviewed multiple times by the Pods team, Felipe Lopes, and certain anonymous personas: smashbopp, rodmono, and gruad. We are deeply thankful for their feedback, repeated edits, brainstorming, and help in formulating our ideas. This work would not have been possible without you.

8 Disclaimer

This paper is for general information purposes only. It does not constitute investment advice or a recommendation or solicitation to buy or sell any investment and should not be used in the evaluation of the merits of making any investment decision. It should not be relied upon for accounting, legal or tax advice or investment recommendations. This paper reflects current opinions of the authors and is not made on behalf of Pods Finance Inc.. The opinions reflected herein are subject to change without being updated.

9 Appendix

1 Options Instrument The Options Instrument component is where the general options contract rules are managed. The Options Instrument can mint, unmint, exercise and withdraw puts and calls, and outlines details like asset pair, strike price, expiration date for each option series. The implementation requires the options to be fully collateralized and the creator can choose between American or European options. If the creator intends to trade the options in the AMM as a secondary market we don't recommend the use of American options since th pricing in the AMM is better suited for European options.

To increase capital efficiency on the collateral locked, the protocol allows users to lock aTokens as collateral and accrued interest in their positions across the options lifetime (this is what we call smart collateral). The options Pods protocol currently has are European and they have a physical settlement, leaving no exposure to liquidation systems.

The Options Protocol is responsible for handling the rules in which the collateral will be managed, exercised, and withdrawn. The trading and pricing facilities take place in the Options AMM. Options tokens are standard ERC20 tokens and can potentially be sold on another DEX or a P2P basis. Anyone can create new options series at any time by following the documentation at https://docs.pods.finance.

2 Core Components The protocol's core components are the Options Instrument and the Options AMM. Currently users can participate:

- as sellers or buyers of either puts or calls in the Options Instrument and/or
- as liquidity providers in the Options AMM.
- create their own options series and equivalent AMM pools. (protocol overview diagram image).

3 Updating User Balance After Withdrawal After calculating how much of each asset a user can withdraw per one's previous open debt with the pool the AMM will calculate what's the total withdrawal amount as follows:

$$A_i = -\left[mAA_i \cdot w_A \cdot \frac{UB_{A_u}}{F_{v_{du}}} + mBA_i \cdot w_B \cdot \frac{UB_{B_u}}{F_{v_{du}}}\right]$$

$$B_i = -\left[mBB_i \cdot w_B \cdot \frac{UB_{B_u}}{F_{v_{du}}} + mAB_i \cdot w_A \cdot \frac{UB_{A_u}}{F_{v_{du}}}\right]$$

And finally update its total asset balance:

$$TB_{A_i} = TB_{A_{i-1}} + A_i$$

$$TB_{B_i} = TB_{B_{i-1}} + B_i$$

Updating its total debt balance:

$$DB_{A_i} = DB_{A_{i-1}} - w_A \frac{UB_{A_u}}{F_{v_{du}}}$$

$$DB_{B_i} = DB_{B_{i-1}} - w_B \frac{UB_{B_u}}{F_{v_{du}}}$$

And updating the user's balance after the withdrawal:

$$UB_{A_u} = UB_{A_{u-1}} \cdot (1 - w_A)$$

$$UB_{B_u} = UB_{B_{u-1}} \cdot (1 - w_B)$$

References

[1] Hull, John. Options, Futures, and Other Derivatives. Pearson, 2018.

[2] Hayden Adams. 2018. url:https://hackmd.io/@477aQ9OrQTCbVR3fq1Qzxg/HJ9jLsfTz?type=view

[3] Rehman, Hassan. 3 May 2021, url:https://xord.com/publications/balancer-v1-balancing-n-dimensions/

[4] Egorov Michael. 10 November 2019, url:https://curve.fi/files/stableswap-paper.pdf

[5] Pintail, Understanding Uniswap Returns. 14 Feb 2019, url:https://pintail.medium.com/understanding-uniswap-returns-cc593f3499ef

[6] Adams, H, N. Zinsmeister, M. Salem, River Keefer, and D. Robinson, Uniswap v3 Core,2021.

[7] Clark J, The Replicating Portfolio of a Constant Product Market with Bounded Liquidity, 3 August 2021 url:https://ssrn.com/abstract=3898384

[8] Anonymous. AAVE Protocol Whitepaper V2.0, January 2020, https://github.com/aave/protocol-v2/blob/master/aave-v2-whitepaper.pdf

[9] Black F, Scholes M, The Pricing of Options and Corporate Liabilities, 1973

[10] Wang Yan. A Well-Posed Algorithm to Recover Implied Volatility, 30 February 2012, url:http://uu.diva-portal.org/smash/get/diva2:506716/FULLTEXT01.pdf

[11] Griffin, John M. and Shams, Amin, Manipulation in the VIX? 23 May 2017, url:https://ssrn.com/abstract=2972979

[12] Lazarev M, Gimaltdinov I, Generalized Leverage Protocol, url:https://static.gearbox.fi/docs/Gearbox.pdf