An Analytical and Empirical Survey of Impermanent Loss

IAN C. MOORE, PHD[1],and JAGDEEP SIDHU, MSC[2]
[1]Syscoin Researcher, Syscoin Platform (e-mail: imoore@syscoin.org)
[2]Syscoin Lead Developer, (e-mail: sidhujag@syscoin.org)

ABSTRACT

Automated market maker (AMM) protocols have played a key role in decentralized finance (DeFi) since Uniswap successfully launched the first constant product market maker in 2018. The popularity of these protocols have since increased to a global market size now measured in billions of US dollars. With this increased demand, there is a need to mitigate losses and maximize yield on these new platforms. In this discussion, we conduct an analytical and empirical survey on the risk profile of these liquidity providers for Uniswap v2, which is commonly known as Impermanent Loss.

INDEX TERMS Impermanent Loss, Liquidity Pool Management, Decentralized Finance, DAOSYS

I. INTRODUCTION

An automated market maker (AMM) protocol is the mechanism used by decentralized exchanges (DEXs) and was first introduced by Uniswap, which was launched on the Ethereum mainnet in November 2018. These DEXs consist of liquidity pools (LPs) represented by various trading pairs (eg. ETH/USDC, ETH/WBTC, etc.) acting as the AMM. Trading activity within these LPs are governed via smart contract through the *constant product trading formula*:

$$xy = k, \tag{1}$$

where x represents the number of tokens for one asset, y represents the number of tokens for the other, and k is a constant, which helps maintain asset price.

When a user places a trade on the DEX, they will typically pay a 0.3% fee which is added to the LP. These cumulated fees are divided proportionally and paid out to all the providers of that particular LP as a yield payment. These collected yields from trading fees provide incentive for liquidity providers to invest in these LPs. This allows LP providers, who would otherwise be regular *hodlers*, to earn additional yield from their assets.

When providers add funds to an LP, the apportioned value to each asset is split 1:1 in USD terms. For instance, say Ethereum is priced at $1,500 per ETH and a provider wants to invest into an ETH/USDC LP, then the provider would have to invest 1 ETH along with an additional $1,500 USDC for a total initial investment value of $3,000 USD. The issue that can arise from investing into these LPs is that there are instances where the USD value of a liquidity provider's stake can decrease (ie, when compared against regular hodling) despite receiving yields from trading fees. This loss is known as Impermant Loss (IL), which was first termed by Pintail in [1], [2]. Many have argued that this term is a bit of a misnormer as the term *impermanent* could create the expectation that losses are subject to a guaranteed reversal, which is not always the case. In 2020, Pintail later updated the term in his original articles to Divergent Loss. However, between that time, the former term had already been adopted by the decentralized finance (DeFi) community. Since this effect is more broadly known as Impermanent Loss, we will be using it for the duration of this discussion.

The motivation for this analysis came out of the need to mitigate losses and maximize yield for a new smart DAO (ie, DAOSYS) that we are setting up on Syscoin's soon to be launched L2 network which is expected to go live mid-to-late 2023 [3]. In this article we provide an analytical and empirical a survey of IL and analytically show that expected impermanent loss is exactly computable and provide simulations to support the analysis. In summary, we provide a general easy-to-follow general guideline for LP managers to best manage their DEXs.

II. ANALYTICAL SURVEY

A. EXPECTED IMPERMANENT LOSS

Impermanent (divergence) Loss is defined as:

$$IL = \frac{V - V_H}{V_H}, \tag{2}$$

where V is the initial value of the portfolio holding, derived as:

$$V = 2L\sqrt{P_0 \alpha}, \tag{3}$$

where L is the liquidity, P_0 is the initial price, α is the price ratio (i.e., $\frac{p_0}{p_f}$), V_H is the held portfolio value, and is derived

by the following:

$$V_H = 2L\sqrt{P_0(1+\alpha)}. \tag{4}$$

Derivations for (3) and (4) are relatively straight forward and can be found in [4], [5], and when applied to the definition of IL in (2) we get:

$$IL = \frac{2\sqrt{\alpha}}{1+\sqrt{\alpha}} - 1. \tag{5}$$

Eq. (5) has been used quite extensively [1], [2], [4], [5] as it provides a way of retrospectively quantifying LP risk expressed in terms of price ratio, α. However, it provides no predictive insight as there is no consideration for historical stochastic price behaviour of a pair of assets. To provide this, we assume that LP price assets behave as a Geometric Brownian Motion (GBM) process, as first discussed in [6]. Using GBM to describe price behaviour has been widely adopted in the financial community; for more context, see Appendix A. When representing IL stochastically, its expectation can be derived as:

$$E\{IL\} = \frac{e^{-\frac{\sigma^2 t}{8}}}{\cosh(\frac{\mu t}{2})} - 1. \tag{6}$$

Eq. (6) provides some predictive insight on the performance of an LP, given an asset's historical price drift, μ, and volatility, σ; see Fig. 1. This expression for the expectation of IL was first found in [6], but with no derivation or explanation on how it was derived, however [7] provides a good elaboration on it's derivation, but with missing steps. For a full analytical derivation of (6), where the remaining steps are provided, see Appendix C.

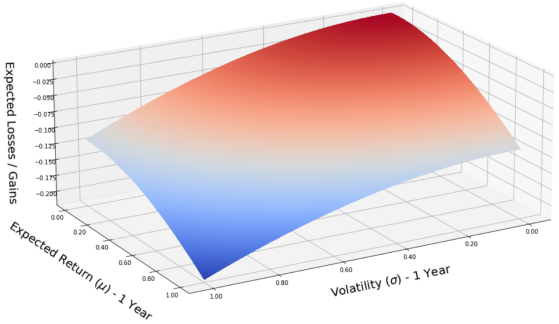

FIGURE 1: Expected impermanent loss for first year (ie, t = 1) using (6); expected IL drops with increasing drift and volatility.

B. EXPECTED PORTFOLIO VALUES

If we consider our assumption that price assets behave as a GBM process, then a portfolio of held assets from (4) is determined by:

$$V_{H,t} = L\sqrt{P_0}(1 + e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}), \tag{7}$$

with an expectation of:

$$E\{V_H\} = 2L\sqrt{P_0}(1 + e^{\mu t}). \tag{8}$$

Likewise, our outside portfolio value of (3) becomes:

$$V_t = 2L\sqrt{P_0}e^{(\frac{\mu}{2} - \frac{\sigma^2}{4})t}e^{\frac{1}{2}\sigma W_t}, \tag{9}$$

with an expectation of:

$$E\{V\} = 2L\sqrt{P_0}e^{(\frac{\mu t}{2} - \frac{\sigma^2 t}{8})}. \tag{10}$$

For the derivation of the expectations in (8) and (10), see Appendix C.

C. EXPECTED RETURNS WITH COMPOUNDING FEES

If we take the aggregation of individual deposits a_i in the form of a geometric mean, this can be expressed by the following:

$$(\prod_{i=1}^{n} a_i)^{\frac{1}{n}} = e^{\frac{1}{n}\sum_i^n \ln a_i}. \tag{11}$$

For sake of brevity, we assume $\beta = \frac{1}{n}\sum_i^n \ln a_i$, hence the growth term becomes:

$$F_t = C_0 e^{\beta t} \tag{12}$$

where β represents constant rate of growth and C_0 represents initial deposit. Hence, we consider (12) to be a simple portfolio growth model. If we consider a dynamic variant of C_0 in the trading fees model of (12), the expectation of (10) becomes:

$$E\{V\} = 2L\sqrt{P_0}e^{(\frac{\mu t}{2} - \frac{\sigma^2 t}{8})}e^{\beta t}. \tag{13}$$

If we consider the expected portfolio value with fee model in (13), and the template outlined in Appendix C, the expected LP returns can be determined as follows:

$$E\{R\} = \frac{e^{-\frac{\sigma^2 t}{8} + z_t}}{\cosh(\frac{\mu t}{2})}. \tag{14}$$

The term e^{z_t} in (14), represents portfolio growth due to the accumulation of trading fees, where z_t can be updated to consider various nuances in trading fee behaviour; see Fig. 2.

III. DEFI PYTHON SIMULATOR

Despite popular perception, treasuries of Decentralized Autonomous Organizations (DAOs) tend to be centrally controlled and do not reflect the true ethos of cryptocurrency (i.e., *not your keys, not your coins*). Syscoin is solving this problem via DAOSYS, which is a new smart DAO protocol technology for DeFi [3], which will be launched sometime in 2023 on Syscoin's L2 Rollux platform.

To realize proper tokenomics design, it is highly inefficient to invest resources into development without first simulating the design to test specifications for various outcomes. This is what every DeFi project in the crypto space is not doing. This is why we are introducing an open source python package to

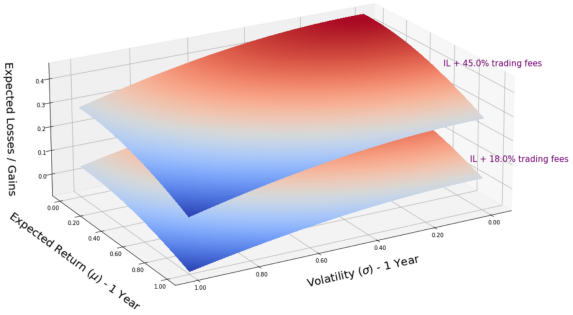

FIGURE 2: Expected impermanent loss with compounding trading fees for first year (ie, t = 1) using (14); increased trading activity results in a upwards shifting of the curve.

simulate various sandboxed DeFi components of DAOSYS so that project engineers, managers and designers can pre-plan outcomes prior to investing valuable resources into development.

With this tool, DAOSYS designers can utilize the plug and play components of our simulator to build tokenomics mockups for business planning purposes so that teams can come together and get collective consensus alongside potential users and investors. Not only is this tool applicable to DAOSYS, it can be used as a general purpose tool to simulate DEX activity for anyone wishing to setup their own LP and wish to stress test their ideas prior to development. This can be used as a powerful design tool to explore the limitations of a DeFi project idea prior to committing valuable resources on development costs (ie, Devs and Project Managers). Table 1 highlights some key components of our DeFi simulator to provide potential users of the system a higher understanding of the package.

Simulator Component	Description
`BrownianModel`	Generate Geometric Brownian motion (GBM) price simulation for a given μ, σ, t, and # of steps; see Appendix A
`SolveDeltas`	Solve swap values Δx_k and Δy_k for a given price change Δp_k; see Appendix D
`Liquidity`	Maintains asset holdings x_k, y_k, Δx_k, Δy_k, liquidity value, price, and fee at time sample k
`LiquidityPoolUSD`	Calculates USD LP values of various forms, given $\{x_k\}_0^N$, $\{y_k\}_0^N$, and $\{p_k\}_0^N$; see Figs. 3 and 4
`EventSelectionModel`	Generates random binomial process representing withdrawal and deposit events, where Pr(deposit) = p, Pr(withdraw) = p-1 and $p \in [0, 1]$
`TokenDeltaModel`	Generates deposit and withdrawals deltas Δx_k and Δy_k randomly sampled from a gamma distribution
`SimulateLiquidity`	Simulates LP activity of swaps, deposits and withdrawals given `TokenDeltaModel`, `Liquidity` objects and pre-generated GBM simulation
`CalcLPFees`	Calculate LP fees (0.3% per swap)

TABLE 1: Descriptions of DeFi python simulator components that was used to generate LP simulations

We have a working beta version of the simulator which is available through Syscoin's Github repository. This simulator is being built alongside DAOSYS, and we are using this to understand the ROI and design considerations for DAOSYS's first usecase (ie, Masternode Yield Farming). Since the simulator is still in the beta stage, the setup is currently not realized to its full intention. However, a usable demo of this tool is available from Syscoin's Github repository for the community to begin using. For a mini python tutorial on how to use this tool, please refer to [12], [13] for a series of example Jupyter notebooks.

The purpose of this tool is primarily for the design aspect of DAOSYS, and is still in its early stages of development. The next stages involve feeding the output of various mathematical models into the simulator framework and to use this tool to simulate other DeFi usecases. This allows us to test a roster of edge cases for building more robust systems. The final goal is to bring this system to a level of maturity so that it can be utilized in parallel with the contracts in real time. Hence, exposing DeFi to a scientific way of design, testing, and implementation.

IV. NUMERICAL SIMULATIONS

In Section II-A we provided the analytic approach to calculating expected IL in (6) and expect returns in (14). In this Section, we provide numerical outcomes using our Defi simulator outlined in Section III to support the analysis.

To simulate LP Activity using GBM price simulations, we implemented the following using the components described in Table 1:

1) Instantiate `Liquidity` object to maintain LP state
2) Instantiate `SimulateLiquidity` object to govern LP simulation events
3) Generate GBM price sequence simulation, $\{p_k\}_0^N$, using `BrownianModel`
4) Using `SimulateLiquidity` object, run LP simulation, for every time sample, k, do:
   a) Solve swap deltas Δx_k and Δy_k using `SolveDeltas` given p_k; see Appendix D
   b) Calculate swap fee, r_k
   c) Run `EventSelectionModel` to determine deposit or withdrawal event with probability $Pr\{deposit\} = 0.55$
   d) Using outcome of previous step, add or subtract liquidity from pool to simulate LP growth or decline
   e) Using `Liquidity` object, update LP state
5) Repeat steps 1-4 `n_path` times

The python code for the above procedure is openly available through Syscoin's Github repository, which can be accessed via [14]. Using this stepwise process, we generate the numerical results which are presented in Tables 2 and 3 for various configurations of drift (μ), volatility, (σ), and sample size (N). These numerical results for expected IL and returns are compared against the theoretical results using (6) and (14) respectively.

We also present simulated temporal LP outcomes from our DeFi simulator as shown in Figs. 3, 4, and 5. Each temporal sample represents the aggregate activity of one time interval which account for typical LP actions such as swaps, deposits and withdraws. It is well-studied that GBM processes are an effective tool to model asset price behavior; see Appendeix A. Thus, activity due to swaps are driven by a set of GBM price sequence paths presented in Fig. 3, and swap deltas Δx, Δy are calculated using the non-linear set of equations presented in Appendix D.

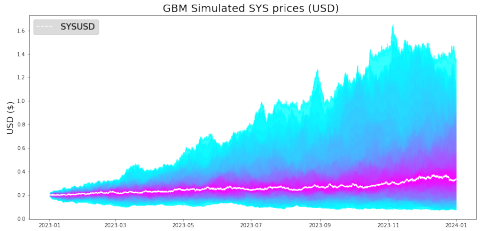

FIGURE 3: Set of simulated geometric Brownian motion (GBM) price sequence, $\{p_k\}_0^N$, paths where $\sigma = 0.5$ and $\mu = 0.4$ representing simulated SYSUSD prices; see Appendix A

LP accumulation in our simulation is acquired via two models: (a) swaps handled by the GBM price sequence; and (b) deposit/withdrawal events driven by a binomial event sequence where Pr(deposit) = 0.55. See the aforementioned stepwise process to see how these processes are generated within our DeFi python simulator. In Fig. 4, we present temporal simulations of LP value growth in USD terms of an initial position of $4,000 USD as represented by the horizontal dashed white line. These charts are sub-divided by LP, fees and total, where the white lines represent estimated expectation at time t. In Fig. 5, we present the temporal LP holdings of SYS, DAI and LP totals, which represent the total aggregate activity of our LP simulation.

		Fees (%)		Imp Loss (%)		Gain / Loss (%)	
μ	σ	Theo.	Sim.	Theo.	Sim.	Theo.	Sim.
0.4	0.5	4.75	4.86	-4.98	-4.79	-0.36	-0.78
		5.15	5.28	-4.98	-4.13	0.04	0.20
		5.33	5.47	-4.98	-3.93	0.22	0.69
0.1	0.1	3.28	3.33	-0.25	-0.29	3.07	2.84
		3.09	3.13	-0.25	-0.22	2.88	2.78
		3.17	3.22	-0.25	-0.21	2.96	2.90
0.8	0.4	5.14	5.28	-9.33	-9.58	-4.55	-6.29
		4.74	4.85	-9.33	-7.33	-4.93	-4.38
		4.78	4.90	-9.33	-6.88	-4.89	-3.82

TABLE 2: Theoretical versus simulated calculations for expected IL and returns; low liquidity (N = 5,000)

V. SUMMARY

In this discussion we perform an analytical and numerical survey of IL for Uniswap v2. If we assume prices behave as a GBM process, we show that IL is exactly computable in

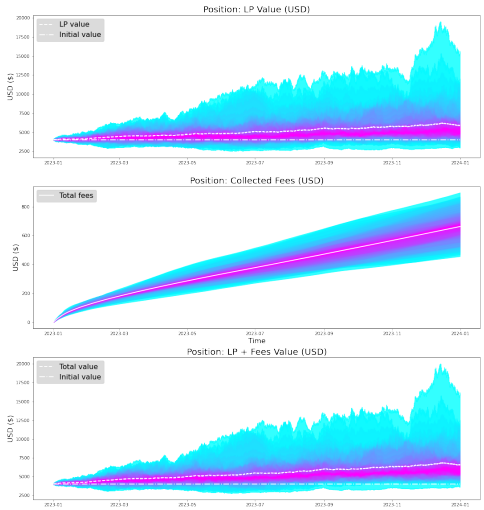

FIGURE 4: Simulation USD values of a LP position with an original value of $4,000 USD (10,000 SYS and $2,000 DIA); (TOP) LP USD value of position holdings, where the horizontal line represents initial investment, and dotted line represents the expectation; (MIDDLE) collected fees from position holdings where dotted line represents the expectation; and (BOTTOM) total USD value of position holdings

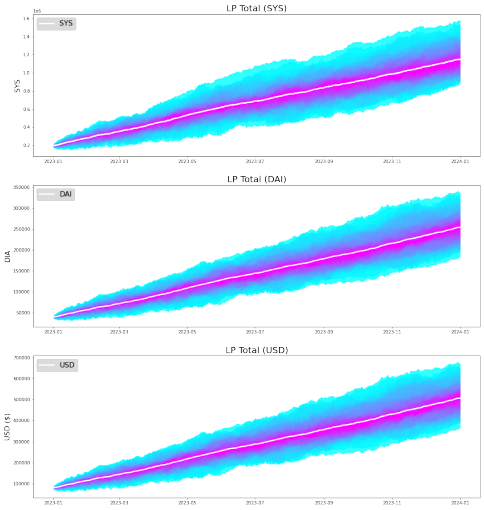

FIGURE 5: Simulated holdings of each asset SYS, DAI and total LP valued in USD within pool. (TOP) LP SYS total; (MIDDLE) LP DAI total; (BOTTOM) LP USD total

closed form (6), and that its computation can be extended to

		Fees (%)		Imp Loss (%)		Gain / Loss (%)	
μ	σ	Theo.	Sim.	Theo.	Sim.	Theo.	Sim.
0.4	0.5	8.82	9.22	-4.98	-5.79	3.78	1.34
		8.72	9.12	-4.98	-6.07	3.68	1.25
		8.49	8.86	-4.98	-3.80	3.43	3.82
0.1	0.1	5.45	5.60	-0.25	-0.23	5.33	5.08
		5.41	5.56	-0.25	-0.19	5.29	5.12
		5.22	5.36	-0.25	-0.27	5.09	4.80
0.8	0.4	8.52	8.89	-9.33	-8.75	-1.27	-3.22
		8.40	8.76	-9.33	-7.81	-1.39	-2.06
		8.67	9.06	-9.33	-7.84	-1.12	-1.99

TABLE 3: Theoretical versus simulated calculations for expected IL and returns; high liquidity (N = 25,000)

	High Liquidity		Low Liquidity	
	volatility (σ)	drift (μ)	volatility (σ)	drift (μ)
Small	2	2	1	1
Medium	0^+	0^+	0	0
Large	-1	-1	-2	-2

TABLE 4: General guideline (ranked by potential profitability) DEXs can apply to their LP management strategy for asset holdings x and y, where μ and σ represent drift and volatility of historical asset prices (ie, $p_k = \frac{y_k}{x_k}$); -2 $\rightarrow$ potential for loss, 0 -> neutral, and 2 $\rightarrow$ potential for gain.

calculate expected returns with compounding fees (14).

We also set up simulations and show that the experimental outcomes also align to the theoretical calculations. The simulator used in this work can be easily extended to include various edge cases to understand risk tolerances aprior to execution which is openly available on Syscoin's Github repos [14], and is the subject of future work.

The theory presented here is very useful for LP managers as we have shown that portfolio risk due to IL can be predetermined through a relatively straight forward back-of-the-envelope calculation, given an asset's historical price drift and volatility. Also, if we have some idea on historical returns due to fees, we have shown that overall expected LP returns can also be estimated in a similar straight forward way. To summarize, we have provided a broad profitability guideline for business managers to follow in Table 4. The fee model that we used is relatively straight forward, and with some additional exploration, the expected returns formula can be updated to better cater to the various forms of LP behavior.

Finally, if long-term profitability is the overall intended goal, these calculations will help remove a lot of the guess work when managing LP risk.

.

APPENDIX A GEOMETRIC BROWNIAN MOTION

Brownian motion (or Wiener Process) is a type of non-stationary stochastic process used to model natural phenomena studied in the Applied Sciences, and is represented as the following distribution:

$$W_t \sim N(0, t), \tag{15}$$

where t represents some time period. A well-known risky asset model for prices, S_t, are represented by the following differential equation:

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{16}$$

where μ represents the drift and σ represents the volatility of the asset. The solution to the asset model of (16) is given by:

$$S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}, \tag{17}$$

which is known as Geometric Brownian Motion (GBM). The idea of using (17) to model stock prices was first discovered by Bachelier [19] and popularized by Samuelson in [18]; details on the derivation of this solution can be found in Chapter 1 of [9].

APPENDIX B MGF OF BROWNIAN MOTION

The moment generating function (MGF), $M(a)$, for a random variable X is $E\{e^{aX}\}$. Thus, the MGF of a normal random variable is given by:

$$\begin{aligned} M(a) &= E\{e^{aX}\} \\ &= \int_{-\infty}^{\infty} e^{ax} f(x) dx \\ &= e^{a\mu + \frac{1}{2}a^2\sigma^2}, \end{aligned}$$

where $X \sim N(\mu, \sigma^2)$. Since a standard Brownian motion process, W_t, is also normally distributed where $W_t \sim N(0, t)$, we have:

$$E\{e^{aW_t}\} = e^{\frac{1}{2}a^2 t}. \tag{18}$$

APPENDIX C EXPECTED IMPERMANENT LOSS DERIVATION

Under the assumption of GBM, equations (3) and (4) can be represented as:

$$V_t = 2L\sqrt{P_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}}, \tag{19}$$

and

$$V_{H,t} = 2L\sqrt{P_0}(1 + e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}). \tag{20}$$

Using the MGF of standard Brownian motion from (18) and (19), we calculate $E\{V\}$:

$$\begin{aligned} E\{V\} &= E\{2L\sqrt{P_0} e^{(\frac{\mu}{2} - \frac{\sigma^2}{4})t} e^{\frac{1}{2}\sigma W_t}\} \\ &= 2L\sqrt{P_0} e^{(\frac{\mu}{2} - \frac{\sigma^2}{4})t} E\{e^{\frac{1}{2}\sigma W_t}\} \\ &= 2L\sqrt{P_0} e^{(\frac{\mu}{2} - \frac{\sigma^2}{4})t} e^{\frac{\sigma^2 t}{8}} \\ &= 2L\sqrt{P_0} e^{(\frac{\mu}{2} - \frac{\sigma^2}{8})}. \end{aligned}$$

Likewise, using the same MGF of standard Brownian motion from (18) and (20), we calculate $E\{V_H\}$:

$$\begin{aligned} E\{V_H\} &= E\{L\sqrt{P_0}(1 + e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t})\} \\ &= L\sqrt{P_0}(1 + E\{e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}\}) \\ &= L\sqrt{P_0}(1 + e^{(\mu - \frac{\sigma^2}{2})t} E\{e^{\sigma W_t}\}) \\ &= L\sqrt{P_0}(1 + e^{(\mu - \frac{\sigma^2}{2})t} e^{\frac{1}{2}\sigma^2 t} \\ &= L\sqrt{P_0}(1 + e^{\mu t}). \end{aligned}$$

Therefore using the the identity $\cosh(x) = \frac{1+e^{2x}}{2e^x}$, and our definition of IL outlined in (2) we have:

$$\begin{aligned} E\{IL\} &= \frac{E\{V_0\} - E\{V_H\}}{E\{V_H\}} \\ &= \frac{2L\sqrt{P_0}e^{(\frac{\mu}{2} - \frac{\sigma^2}{8})}}{L\sqrt{P_0}(1+e^{\mu t})} \\ &= \frac{2e^{(\frac{\mu}{2} - \frac{\sigma^2}{8})}}{1+e^{\mu t}} - 1 \\ &= \frac{e^{-\frac{\sigma^2 t}{8}}}{cosh(\frac{\mu t}{2})} - 1. \end{aligned}$$

APPENDIX D SOLVE LP SWAP DELTAS

Here we address the problem of updating asset balances (ie, x_k,y_k) upon each price change Δp_k, where $\{p_k\}_0^N$ is a simulated GBM process representing the prices of our asset (valued in USD) over time sample k. Price change, Δp_k, in a liquidity pool are determined as:

$$\begin{aligned} \Delta p_k &= \frac{y_k}{x_k} - \frac{y_{k-1}}{x_{k-1}} \\ &= \frac{\Delta y_k + y_{k-1}}{\Delta x_k + x_{k-1}} - \frac{y_{k-1}}{x_{k-1}}, \end{aligned}$$

and asset trade price is determined as:

$$p_k = \frac{\Delta y_k}{\Delta x_k}. \tag{21}$$

Thus, given the above equations, we derive the following system of non-linear equations:

$$\frac{x_{k-1}\Delta y_k - \Delta x_k y_{k-1}}{x_{k-1}^2 + x_{k-1}\Delta x_k} - \Delta p_k = 0 \tag{22}$$

$$\frac{\Delta y_k}{\Delta x_k} - p_k = 0. \tag{23}$$

Since we have the previous balances y_{k-1}, x_{k-1}, price change Δp_k, and the current price p_k from our simulated GBM, we can solve for the swap balances Δx_k and Δy_k under the constraints $\Delta y_k > 0$ and $\Delta x_k > 0$ for every k. To achieve this, we used the `optimize.fsolve` function from `scipy` package in python.

REFERENCES

[1] Pintail, *Uniswap: A Good Deal for Liquidity Providers?*, Jan. 2019. Accessed on: Dec 2022. [Online]. Available: https://pintail.medium.com/uniswap-a-good-deal-for-liquidity-providers-104c0b6816f2

[2] Pintail, *Understanding Uniswap Returns*, Feb. 2019. Accessed on: Dec 2022. [Online]. Available: hhttps://pintail.medium.com/understanding-uniswap-returns-cc593f3499ef

[3] C. Doge, I. Moore, R. Arbour, and J. Sidhu, *DOASYS, Smart DAO Protocol for Decentralized Finance*, Syscoin Github repos, Oct. 2022, Accessed on: Mar 2023. [Online]. Available: https://github.com/syscoin/daosys/blob/main/latex/whitepaper/article.pdf

[4] P. Erins, *How to calculate Impermanent Loss: full derivation*, Jun. 2021. Accessed on: Dec 2022. [Online]. Available: https://medium.com/auditless/how-to-calculate-impermanent-loss-full-derivation-803e8b2497b7

[5] A. Aigner and G. Dhaliwal, *UNISWAP: Impermanent Loss and Risk Profile of a Liquidity Provider*, Jun. 2021. Accessed on: Dec 2022. [Online]. Available: https://arxiv.org/pdf/2106.14404.pdf

[6] G. Lambert, *Calculating the Expected Value of the Impermanent Loss in Uniswap*, Oct. 2021. Accessed on: Dec 2022. [Online]. Available: https://lambert-guillaume.medium.com/an-analysis-of-the-expected-value-of-the-impermanent-loss-in-uniswap-bfbfebbefed2

[7] Danr, *Expected Impermanent Loss in Uniswap V2 & V3*, Mar. 2022. Accessed on: Dec 2022. [Online]. Available: https://medium.com/gammaswap-labs/expected-impermanent-loss-in-uniswap-v2-v3-7fb81033bd81

[8] Danr, *Total Returns and Impermanent Loss in Uniswap V2*, Dec. 2021. Accessed on: Dec 2022. [Online]. Available: https://medium.com/gammaswap-labs/total-returns-and-impermanent-loss-in-uniswap-v2-9f3d5b6ebc89

[9] N. Privault, *Notes on Financial Risk & Analytics*, Jan 2023, Accessed on: Feb 2023. [Online]. Available: https://personal.ntu.edu.sg/nprivault/MH8331/financial_risk_analytics.pdf

[10] Uniswap Blog, *A short history of Uniswap*, Sept. 2019. Accessed on: Sept 2022. [Online]. Available: https://uniswap.org/blog/uniswap-history

[11] I.C. Moore, *Masternode Yield Farming as a DAOSYS Usecase: Part 1*, Medium Article, Aug. 2022. Accessed on: Sept 2022. [Online]. https://icmoore.medium.com/masternode-yield-farming-as-a-daosys-usecase-part-1-8485ab7ba721

[12] I.C. Moore, *Masternode Yield Farming as a DAOSYS Usecase: Part 2*, Medium Article, Aug. 2022. Accessed on: Sept 2022. [Online]. https://icmoore.medium.com/masternode-yield-farming-as-a-daosys-usecase-part-2-8906d0a11c27

[13] I.C. Moore, *Simulation: Masternode Yield Farming*, Syscoin Github repos, Accessed on: Apr 2021. [Online]. Available: https://github.com/syscoin/daosys/tree/main/notebooks/simulation

[14] I.C. Moore , *Simulation: LP Activity using GBM*, Syscoin Github repos, Accessed on: Apr 2021. [Online]. Available: https://github.com/syscoin/daosys/tree/main/notebooks/research/impermanent_loss

[15] Syscoin News, *Introducing Rollux, Syscoin's Rollup Suite Ready to Take Market by Storm*, Syscoin, Jun. 2022. Accessed on: Sept 2022. [Online]. Available: https://syscoin.org/news/introducing-rollux-syscoins-rollup-suite-ready-to-take-market-by-storm

[16] J. Sidhu, *A Design For An Efficient Coordinated Financial Computing Platform*, Feb 2021, Accessed on: Sep 2021. [Online]. Available: https://jsidhu.medium.com/a-design-for-an-efficient-coordinated-financial-computing-platform-ab27e8a825a0

[17] J. Sidhu and I.C. Moore, *Syscoin 4: A Peer-to-Peer Electronic Cash System Built For Decentralized Web 3.0 Business Applications*, Syscoin Platform, Dec. 2021. Accessed on: Sept 2022. [Online]. Available: https://syscoin.org/$syscoin4\_whitepaper.pdf$

[18] P. Samuelson, *Rational Theory of Warrant Pricing*, Industrial Management Review (pre-1986), Cambridge Vol. 6, Iss. 2, (Spring 1965): 13.

[19] L. Bachelier, *Théorie de la spéculation*, Annales scientifiques de l'École Normale Supérieure, Série 3, Tome 17 (1900), pp. 21-86.