COS/ECE 473

Elements of Decentralized Finance

Investigating incentive compatible AMMs in binary prediction markets[1]

Authors:

Arya Maheshwari arya@princeton.edu
Janum Shah jpshah@princeton.edu
Abiram Gangavaram abiramg@princeton.edu

2022-2023 – Spring Semester

[1]The code for this project can be found in our GitHub repository here. The PredictionMarkets.ipynb notebook contains our Python simulation code and our smart contract Solidity code pasted in, while the `solidity` folder contains our smart contract code and Truffle testing suite.

Contents

1 Introduction

1.1 Prediction Markets

Prediction markets are financial markets where participants can trade securities corresponding to the outcomes of prediction markets. In this work, we focus on *binary* prediction markets, in which exactly one outcome will occur, meaning only a single security out of the set of possibilities will eventually pay out. Prediction markets rely on *scoring rules* to quantify how to reward participants for the accuracy of their bets; this means that scoring rules should be designed to incentive participants to trade truthfully. Scoring rules that adhere to this property are *incentive compatible* or *strictly proper* [12].

1.2 Why in DeFi?

The former CTO of Coinbase, Balaji Srinivasan, claimed that "Blockchain-based prediction markets may be the one force strong enough to counterbalance the spread of incorrect information on social media. They give people a financial incentive to seek the truth and then protect them with the twin shields of pseudonymity and decentralization" [1]. Decentralized prediction markets can help mitigate the spread of misinformation if they can properly incentivize market participants to trade truthfully. Unlike traditional markets, where one centralized party controls the pricing, decentralized markets allow market participants to collectively price assets automatically based on their beliefs. This means decentralized prediction markets are faster, cheaper, and more secure to operate. From a participant's point of view, they are also more fair and transparent.

Decentralized prediction markets leverage Automated Market Makers (AMMs) to continuously provide liquidity to the market. Generally, any market can suffer from a lack of liquidity, and in many financial markets, the responsibility to keep the market liquid falls on large financial institutions. With AMMs, however, asset prices are continuously priced using specific pricing algorithms, meaning that participants can always interact with the market when they want to. AMMs are the key driver of a fair and transparent model in decentralized prediction markets.

1.3 Our Work

We are primarily interested in investigating the behavior of AMMs based on different scoring rules. In this work, we build a simulation for a decentralized prediction market powered by an AMM. We implement the Logarithmic Market Scoring Rule (LMSR) [6] and its more sophisticated variant, the Liquidity Sensitive Logarithmic Market Scoring Rule (LS-LMSR), introduced in [10], and compare their behavior across various metrics.

This work is important in understanding the benefits and drawbacks of various scoring rules for those interested in deploying their own prediction markets. dApps like Augur [8] and Gnosis [7] have already built platforms for easy-to-deploy prediction markets, and the popularity of these platforms is only growing. As such, this work is fundamental for traders and market owners who are looking to get a better, more comprehensive understanding of how scoring rules power AMMs and their differing effects in practice.

2 Theory

2.1 Scoring Rules

We begin by formalizing the theoretical underpinnings of the specific AMMs we aim to investigate in the contet of prediction markets. The starting point for this is the concept of a *scoring rule*: a function that maps a participant's reported belief vector $\mathbf{x}$ on the outcome that will occur (a probability distribution over the n possible market outcomes) and the final outcome i to a payout. Mathematically, a scoring rule is a function S s.t. $S(\mathbf{x}, i) \in \mathbb{R}$.

To elicit the best information, we want to design scoring rules that incentivize honest reports. This desired property leads to the definition of a *strictly proper* scoring rule, a scoring rule $S(\mathbf{x}, i)$ in which a participant's expected payoff is maximized if they report their true belief $\mathbf{p}$: mathematically, such that $\mathbf{p} = \arg\max_{\mathbf{x}} \mathbb{E}_{i\sim\mathbf{p}}[S(\mathbf{x}, i]$. Among the most widely studied strictly proper rules due to its simple form is the *logarithmic scoring rule*, defined by $S(\mathbf{x}, i) = \log x_i$.

The first step towards implementing a scoring rule $S(\mathbf{x}, i)$ in real-world prediction markets is to adapt it into a *market scoring rule* (MSR), a construction that can handle multiple market participants. Specifically, in a market with multiple traders acting sequentially, at each time step j, a MSR maintains a current prediction $\mathbf{x}$ (initialized arbitrarily) and asks trader j to report their prediction $\mathbf{x}_j$, then updating $\mathbf{x} \leftarrow \mathbf{x}_j$ for time step $j + 1$. When an outcome i is eventually realized, the trader at each time step j receives payout $S(\mathbf{x}_j, i) - S(\mathbf{x}_{j-1}, i)$ [12]. For example, this construction is known as the logarithmic market scoring rule (LMSR) when the logarithmic scoring rule is used as $S(\mathbf{x}, i)$ [6].

2.2 Construction of Automated Market Makers

The key idea that bridges MSRs and real-world prediction markets is that prediction reporting in an MSR can be implemented in an equivalent yet more practical way by designing an Automated Market Maker (AMM) with a specific price function, where traders now buy/sell shares to update the market prediction but face the same incentives as if they were interacting with a MSR [4, 5, 12].

Recall that AMMs allow traders to continuously interact with the market by setting a *price function* for buying/selling shares. We represent the state of the market through a *quantity vector* $\mathbf{q}$ (initialized to some starting $\mathbf{q_0}$) where each component q_i represents the net shares of outcome i that have been bought. The AMM price function is then a map $\mathbf{p}(\mathbf{q})$ from $\mathbf{q}$ to a vector of prices, where $p_i(\mathbf{q})$ is the price for a share for outcome i. The cost of executing a trade $\mathbf{q}'$ starting from state $\mathbf{q_0}$ is given by

$$\int_0^{\mathbf{q}'} \sum_{i=1}^{n} p_i(\mathbf{q}_0 + \mathbf{x}) d\mathbf{x}.$$

Then, the idea is that to implement a MSR based on some $S(\mathbf{q}, i)$, we can set the price function $\mathbf{p}(\mathbf{q})$ accordingly such that traders face the same incentives with the AMM as in the MSR. Formally, for any trade $\mathbf{q}'$ starting from state any $\mathbf{q_0}$, and any eventual outcome i, the eventual profit from $\mathbf{q}_0 \to \mathbf{q}_0 + \mathbf{q}'$ should be equal to the profit from updating from $\mathbf{p}(\mathbf{q}_0) \to \mathbf{p}(\mathbf{q}_0 + \mathbf{q}')$ in the MSR. (Note the idea of market prices being interpreted as probabilities: we view $\mathbf{p}$ as a normalized price here.)

An example of this for the LMSR is demonstrated in Theorem 5.1 in [12]. Specifically, an AMM based on the LMSR (henceforth the "LMSR AMM") has price $p_i(\mathbf{q}) = \frac{e^{q_i/b}}{\sum_{j=1}^{n} e^{q_j/b}} = \frac{\partial C(\mathbf{q})}{\partial q_i}$, where $C(\mathbf{q}) = b \ln \sum_{j=1}^{n} e^{q_j/b}$ is the *cost function* (for some liquidity parameter b).

2.3 Desirable Properties in AMMs

The following are three main properties we would want when designing an AMM [10]:

1. **Path independent**: If the cost of a transaction is determined only by the initial and final market states, not the path taken in between. Without path independence, an AMM will be vulnerable to arbitrage: a "money pump" where a trader can go from state $\mathbf{q}_1 \to \mathbf{q}_2 \to \mathbf{q}_1$ using different paths and end up with a profit.
2. **Translation invariant**: If the prices sum to one, i.e. $\sum_i p_i(\mathbf{q}) = 1$. Among a few other useful properties, an important practical consequence is that prices can be directly interpreted as current market beliefs of probabilities of outcomes.
3. **Liquidity sensitive**: If price elasticity adjusts based on the volume of market activity. Formally, an AMM is liquidity *insensitive* if $p(\mathbf{q} + k\mathbf{1}) = p(\mathbf{q})$ for all k, $\mathbf{q}$. The idea is that a liquidity sensitive market is more realistic than a liquidity insensitive one, as a trade of some fixed amount should influence prices less in a more liquid market than a less liquid one.

Note that path independence automatically holds if the price function $p_i(\mathbf{q}) = \frac{\partial C(\mathbf{q})}{\partial q_i}$ (i.e. can be represented as the gradient of some cost function); then, the cost of a transaction $\mathbf{q}_1 \to \mathbf{q}_2$ to the trader is always $C(\mathbf{q_2}) - C(\mathbf{q_1})$ (over any path).

The LMSR AMM satisfies properties 1 and 2 but not property 3. In fact, Othman et. al prove in [10] that it is impossible for any AMM to satisfy all three properties.

2.4 A Liquidity-Sensitive LMSR AMM

The AMM that is the focus of our investigation is the *liquidity sensitive LMSR* (LS-LMSR) AMM proposed by Othman et al. [10], a modification of the LMSR AMM that relaxes translation invariance in favor of liquidity sensitivity. The LS-LMSR AMM is defined by the cost function

$$C(\mathbf{q}) = b(\mathbf{q}) \ln \sum_{j=1}^{n} e^{q_j/b(\mathbf{q})}$$

where $b(\mathbf{q}) = \alpha \sum_i q_i$ for some parameter α. Specifically, the fixed b parameter of the LMSR now becomes a function of market quantity, which is what makes this AMM liquidity sensitive — such that the price of a fixed-size transaction decreases as $\mathbf{q}$ increases in magnitude (see Figures 2 and 3 of [10]).

According to the theory presented in [10], the LS-LMSR offers benefits both as a more realistic model that responds to liquidity changes but also in terms of final AMM revenue. The LMSR AMM is expected to run at a loss as long as final market prices are more accurate than initial ones (viewed as the "price of information" in prediction markets); meanwhile, the

LS-LMSR is expected to have less loss than LMSR, even yielding a profit in a range of final conditions. [2]

Our project goal is thus to investigate the performance and properties of the LS-LMSR AMM compared to the LMSR AMM in practice, via a simulation of prediction markets based on both of these AMMs, and to understand how it may fit into the broader DeFi context by implementing a smart contract and looking into various applications.

3 Approach

3.1 Overview

To investigate the characteristics of the LS-LMSR AMM applied to a binary prediction market compared to those of an LMSR AMM, we conducted a simulation-based study in Python. By iterating on an initial simulation design using traders with perfect information, we ultimately designed a more realistic Noisy Information Market, more accurately modeling each trader's beliefs as a weighted average of (1) "private information" drawn uniformly from an interval around the ground truth and (2) an average of previous market beliefs. We update the weighting over time so traders are increasingly influenced by market perception as time goes on, representing a converging market belief. We varied the ground truth distributions and the number of traders to understand how different events behave in both markets, and we also varied α for the LS-LMSR AMM to investigate the impact of the hyperparameter on the AMM's performance and accuracy.

3.2 Modeling a Prediction Market

We first need to be able to set up a model for simulating a general prediction market. While designing an reasonable simulation, we identify a set of realistic properties that we wanted our market simulation to adhere to:

1. Each event has a *ground truth* probability for each outcome.
2. All the traders' personal beliefs before participating in the market resemble some reasonable distribution surrounding this ground truth (but in general, they don't exactly know the ground truth).
3. Traders' beliefs periodically update to reflect the current market's beliefs. That is, each trader's belief should be influenced to some degree by the overall market.
4. Market participants act optimally; if their expected value from trading an outcome (based on current their individual belief) is positive, they will always continue to trade that outcome.
5. Over time, the market beliefs should more or less converge (rather than diverge or randomly oscillate).

[2] We note that since the original LS-LMSR paper, other scoring rules based on the LS-LMSR have recently been developed — such as Zeitgeist's Rikkido scoring rule [11], which aims to make LS-LMSR respond dynamically to market volatility and is discussed further in Section 5.3 and Appendix B.

6. At the end of each trial, the actual outcome should be realized by drawing from the initial ground truth distribution.

Collectively, these properties guide our design of the noisy information prediction market simulation we present below.

3.3 Noisy Information Simulation

To compare a LMSR AMM and a LS-LMSR AMM in our Noisy Information Simulation, we start by setting up two prediction markets, one for LMSR and one for LS-LMSR, for a sequence of N traders to interact with. We focus on $n = 2$ outcome markets for simplicity (so we represent the ground truth probability distribution with the probability p of outcome 1 occurring).

Algorithm 1 presents a detailed description of our noisy market simulation procedure for a given market. In each simulation trial, we first initialize each market with an initial quantity q_0, a ground truth probability p, and the α parameter for LS-LMSR or b parameter for LMSR. For each trader t (acting in round t) in the sequence of N traders, an initial belief x_t is drawn uniformly from an interval around p. Then, the idea is that each trader t will update their belief to be a weighted average of this initial belief and the "market belief" at round t, which we model as the average (normalized) market price across a recent period of rounds. Furthermore, we employ a variable weighting scheme such that traders in later rounds will weight the market belief more heavily than traders in earlier rounds. The motivation for these design choices is that to model how real-world traders use a mix of private and market information and how real-world market prices converge and get more accurate as more traders participate.

With the beliefs thus determined, the trader t will then pick an outcome i and buy shares of that outcome while their expected value in doing so is positive, and then we move on to the next round/trader $t + 1$. Note that to evaluate expected value, traders use the price $p_i(\mathbf{q})$ and their actual belief, where we assume they act truthfully because we base the AMMs on a strictly proper scoring rule. At the end, we (1) compute the final market price accuracy (interpreting the prices as probabilities and comparing to the ground truth p); (2) compute the expected AMM revenue using the AMM's cost function; and for some experiments (3) actually draw an outcome and compute the resulting AMM revenue.

Note: To enable a fair basis for comparison between the two markets in a given trial, when a given α is used in the LS-LMSR AMM, we set the b parameter of the corresponding LMSR AMM such that the *worst-cases losses* of both markets are the same (following a similar practice in [10]), and in a given trial the ground truth probabilities p in each market are the same. Furthermore, we also have the same N traders interact with each market (i.e. in a given round t, the traders interacting with each market will have the same behavior model and same *initial* belief x_t).

Overall, this design enables us an expressive noise-modeling framework that has the capacity to model many aspects of trader noise and information relevant to real-world markets, which also can be fine-tuned and experimented with for testing AMMs in future studies. Out of the parameters used in the simulation (see Algorithm 1), the following remain fixed throughout our experiments: we always set initial quantity $q_0 = [100, 100]$, lookback $\ell = \frac{N}{4}$ (a quarter of the

Algorithm 1 Noisy Information Simulation

Parameters:
(1) α for LS-LMSR or b for LMSR; (2) initial quantities $q_0 = [100, 100]$
(3) number of traders N; (4) ground truth range $Y = [y_1, y_2]$; (5) private noise σ;
(6) lookback $\ell := \frac{N}{4}$; (7) weight bounds k_{min}, k_{max}; (8) buy fraction Δ

Draw $p \leftarrow Y$ uniformly and initialize market with q_0, α or b (LS-LMSR vs LMSR), and p
for each round t s.t. $1 \leq t \leq N$ **do**
  Initialize Trader t with belief $x_t \leftarrow [p - \sigma, p + \sigma]$ uniformly ▷ Nb: ensure x_t within $[0.01, 0.99]$
  $k \leftarrow k_{max} + (k_{min} - k_{max})\frac{i-1}{N-1}$ ▷ Variable weighting scheme
  Set market average $m =$ average $p_1(\mathbf{q})$ across last ℓ rounds
  $x_t \leftarrow kx_t + (1-k)m$; set $belief(1) = x_t$, $belief(2) = 1 - x_t$ ▷ Set belief as weighted average
  Select outcome i s.t. $p_i(\mathbf{q}) < belief(i)$ (else $i =$ NONE) ▷ Buy while positive expected value
  while i is not NONE **do**
    Trader t buys outcome i, amount $\Delta^* q_i$: $q_i \leftarrow (1 + \Delta) * q_i$ ▷ Iteratively buy small increments
    Set $i =$ NONE if $p_i(\mathbf{q}) \geq belief(i)$
  end while
end for
End simulation: compare p to final market price $p_1(\mathbf{q})$ for final market price accuracy
Calculate AMM expected revenue based on initial $\mathbf{q}_0$, final $\mathbf{q}$, ground truth probability p
Draw final outcome j according to ground truth distribution $[p, 1-p]$ to determine actual revenue

Outputs: Final market price accuracy, AMM expected revenue (& optionally: outcome, actual revenue)

total number of traders); weight bounds $k_{min} = 0.2$, $k_{max} = 0.6$; buy fraction $\Delta = 0.02$; and private noise $\sigma = 0.2$.[3]

4 Results

Figure 1 illustrates how final market price accuracy varies with the number of traders N in our Noisy Information Simulation, shown for $\alpha = 0.03$ in the LS-LMSR (and corresponding $b = 150.27$ in the LMSR) and ground truth p drawn from $Y = [0.6, 0.8]$. Specifically, we see that in both AMMs, accuracy increases as the number of traders does. This provides a good sanity check that our simulation correctly models one of the key principles we laid out based on real-world markets—that when more traders participate, the market converges towards a more accurate prediction.

Figure 2 plots the final price accuracy and expected AMM revenue across 100 trials of the simulation with $\alpha = 0.03$ and $N = 20$, illustrating the tradeoff that exists between these quantities. In particular, we see that accuracy decreases as expected revenue increases and that the LS-LMSR AMM incurs less of a cost practice than the LMSR AMM (corroborating the theory presented by [10]) but yields worse accuracies. Intuitively, an inverse correlation between profit and accuracy makes sense: the AMM extracts more profit from traders when they bet farther from the ground truth. In simpler terms, the farther a trader's belief is from the ground truth, the more the AMM can capitalize on their loss. Refer to Figure 6 in the Appendix for results showing *realized* revenue (rather than expected) against final accuracy.

[3]For a fixed weighting scheme rather than a variable one, simply set $k_{min} = k_{max}$ to the desired fixed value.

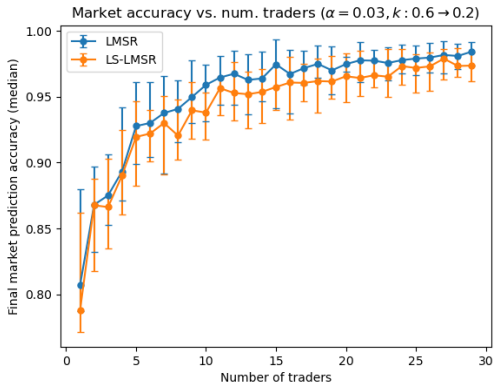


Figure 1: Final price accuracy for LMSR and LS-LMSR revenue for simulations with varying numbers of traders. $\alpha = 0.03$, and ground truth p drawn uniformly from $[0.6, 0.8]$.

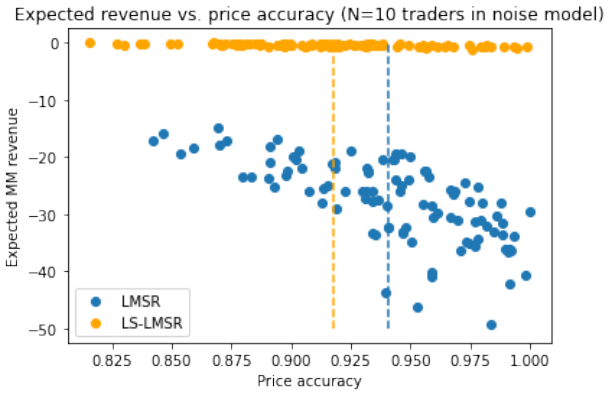


Figure 2: Expected revenue vs. price accuracy for both LMSR and LS-LMSR over 100 trials, $\alpha = 0.03$. Ground truth p drawn from $(0.1, 0.2)$. Orange and blue vertical lines show mean accuracy for LS-LMSR and LSMSR, respectively.

Figures 3 and 4 present the resulting final price accuracies and expected AMM revenue across a range of LS-LMSR α values, again with corresponding LMSR b values set to equalize worst-case-losses (for $N = 20$ traders). Figure 3 shows that LMSR accuracy stays near-perfect across increasing α while LS-LMSR accuracy steadily drops. A possible explanation is that increasing α in LS-LMSR more strongly penalizes later traders (when market quantities $\mathbf{q}$ are higher) with higher prices, compared to increasing b (not dependent on $\mathbf{q}$) in LMSR, thus deterring traders from making reports and resulting in worse accuracy with LS-LMSR than LMSR.

Interestingly, Figure 4 shows that AMM revenue remains roughly constant for both LS-LMSR and LMSR across α. The LS-LMSR theory presented in [10] suggested broadly that a higher α could be interpreted as a higher commision (i.e. yielding higher revenue) for the AMM. However, this claim was based on their derivation of the theoretical upper bound on the sum of prices given by $1 + \alpha n \log n$ (section 4.2 in [10]), which was then interpreted as implying that LS-LMSR AMM revenue would in theory increase with α; the authors did note that effects in practice might be ambiguous due to higher prices deterring trades. Our empirical work thus provides evidence that increasing α indeed has competing effects that in practice cancel out in terms of effect on AMM revenue.

5 DeFi Applications

We now pivot to investigating applications of prediction markets and AMMs based on these scoring rules in the broader DeFi context.

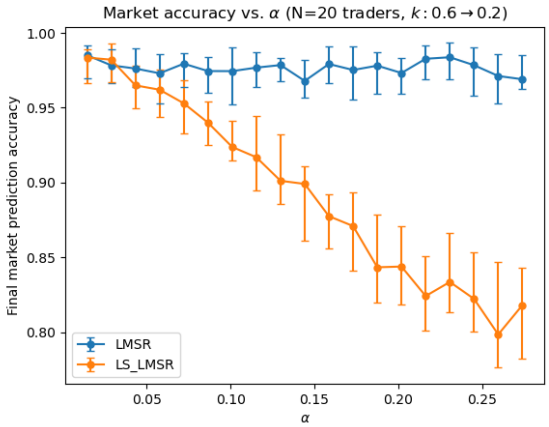


Figure 3: Final price accuracy for LMSR and LS-LMSR revenue for simulations across α. Number of traders set to 20; ground truth p drawn uniformly from $[0.6, 0.8]$.

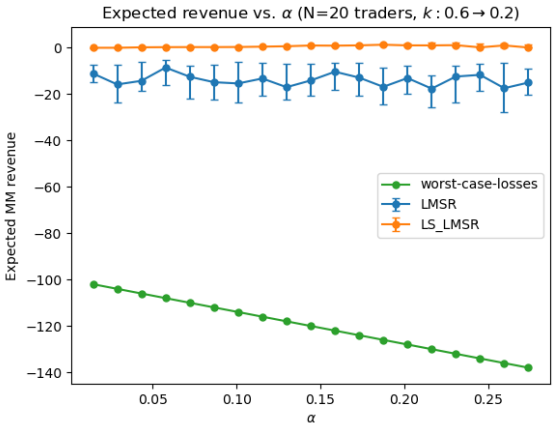


Figure 4: Expected revenue for LMSR and LS-LMSR revenue for simulations across α, along with worst-case loss. Number of traders set to 20; ground truth p drawn uniformly from $[0.6, 0.8]$.

5.1 Smart Contract Development in Solidity

Our inquiry started with our own implementation of a prediction market in Solidity. This exercise not only helped us gain firsthand experience of the intricacies involved in developing a prediction market but also the mechanics of the underlying DeFi protocols.

Our smart contract supports the creation of both LS-LMSR and LMSR-based markets, with specified α and b parameters. To trade an outcome, traders can query the current cost of a specific trade and then buy shares of a certain outcome by submitting bets. They can redeem their winnings if the outcome is realized. Additionally, traders can query the current prices and quantities of different outcomes to observe the current state of the market.

The primary difficulty in building this contract was the implementation of the scoring logic in the cost and price functions. This was due to the lack of existing support for floating-point numbers in Solidity. We circumvented this by utilizing a third-party library, PRBMath [2]. However, the floating-point logic in the library drastically increased gas prices and we would like to explore more gas-efficient alternatives in the future.

We developed our smart contract in Remix and utilized Truffle and Web3.js to test before deploying. The Truffle testing suite and configuration can be found in our GitHub here. As a demonstration, we have deployed both LMSR and LS-LMSR versions of our smart contract (for $n = 2$ items, $\alpha = 0.03$ for LS-LMSR and $\beta = 150.27$ for LMSR like in our simulations), linked here: LMSR contract and LS-LMSR contract.

5.2 Augur and Gnosis

In addition to our own prediction market implementation, we explored two existing DeFi prediction markets: Augur and Gnosis. Augur is a decentralized prediction market platform that is built on Ethereum. Users can leverage Augur's smart contracts to both participate in and create prediction markets mapped to specific events. In markets created with Augur,

the underlying AMM is scored using LS-LMSR [8]. Gnosis is similar to Augur, but offers two choices of AMMs: an LMSR-based AMM and a Constant Product Market Maker (CPMM) [7].

5.3 Zeitgeist PM and the Rikkido Scoring Rule

While researching existing DeFi prediction markets, we encountered the novel "Rikkido" scoring rule proposed by Zeitgeist PM [11]. The Rikkido scoring rule takes the LS-LMSR and adapts it based on recently proposed ideas on Dynamic Market Making [9], which allows the AMM to dynamically adjust its parameters based on market volatility. The key idea is to modify the cost function to discourage trades in high volatility periods and encourage trades during low volatility periods; the Rikkido scoring rule thus combines the robust structure of LMSR, the liquidity sensitivity of LS-LMSR, and the flexibility of dynamic market making. After reaching out to Zeigeist developers to better understand this emerging rule, we found that the implementation of Rikkido-based markets in Zeitgeist PM is still in development. We subsequently conducted a preliminary numerical study of Rikkido, described in Appendix B.

6 Conclusion and Future Work

6.1 Discussion

As the relevance of DeFi prediction markets grows, it's important to have a comprehensive understanding of the scoring rules behind various AMMs we use as both traders and market owners. In this work, we see that LMSR and LS-LMSR are both high-quality options on which to base AMMs, but depending on a market owner's priorities, one may be more performant than the other. This work, however, only touches the surface of investigation into this field. The growing marketplace of DeFi prediction markets warrants even more research in this area.

6.2 Future Work

A first avenue for future work involves developing on the simulation and noisy prediction market model we present. Future investigations could experiment with more sophisticated market/trader models and also different market conditions: while we draw a static ground truth at the start, it would be interesting to investigate how the different AMMs perform in markets with (1) an "information shock" (representing an event that drastically changes outcome probabilities) or (2) a ground truth that moves according to a random walk.

Another area of future work would be to employ our simulation framework to explore the various types of scoring rules and AMMs that have been developed since the LMSR and LS-LMSR. For instance, our brief analysis in Appendix B could launch a much deeper study of the Rikkido scoring rule [11], and another AMM model that would be interesting to investigate is the Bayesian Market Maker [3].

Finally, one limitation of our smart contract implementation is its gas inefficiency. The development of a more efficient floating-point library could greatly decrease gas costs. While we found PRBMath [2] to be the most efficient existing open-source Solidity floating-point library, improvements to the natural log and exponential functions could increase the gas efficiency of our contract.

7 Code

The code for this project can be found in our GitHub repository here. The PredictionMarkets.ipynb notebook contains our Python simulation code and our smart contract Solidity code pasted in, while the `solidity` folder contains our smart contract code and Truffle testing suite.

References

[1] Balaji Srinivasan (@balajis). Blockchain-based prediction markets. https://twitter.com/balajis/status/1087069353688715264?lang=en, Jan. 2019.

[2] Paul Berg. Prbmath. https://github.com/PaulRBerg/prb-math.

[3] Aseem Brahma, Mithun Chakraborty, Sanmay Das, Allen Lavoie, and Malik Magdon-Ismail. A Bayesian Market Maker. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, page 215–232, New York, NY, USA, 2012. Association for Computing Machinery.

[4] Yiling Chen and David M. Pennock. Designing markets for prediction. *AI Magazine*, 31(4):42–52, Dec. 2010.

[5] Yiling Chen and David M. Pennock. A utility framework for bounded-loss market makers. *CoRR*, abs/1206.5252, 2012.

[6] Robin Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1, May 2003.

[7] Gnosis Ltd. Gnosis: The community-run chain. https://www.gnosis.io/.

[8] PM Research LTD. Augur: Your global, no-limit betting platform. https://augur.net/.

[9] Andrew Nguyen, Loi Luu, and Ming Ng. Dynamic Automated Market Making. https://files.kyber.network/DMM-Feb21.pdf, Feb. 2021.

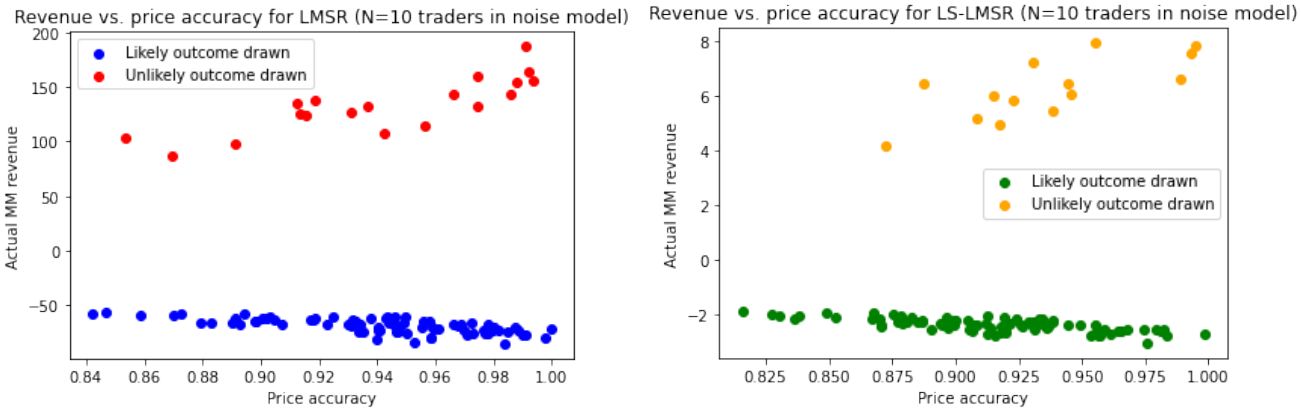
[10] Abraham Othman, David M. Pennock, Daniel M. Reeves, and Tuomas Sandholm. A practical liquidity-sensitive automated market maker. *ACM Trans. Econ. Comput.*, 1(3), Sep. 2013.

[11] Numa De Pablo. Introducing Zeitgeist's "Rikiddo Scoring Rule". https://medium.com/zeitgeistseer/introducing-zeitgeists-rikiddo-scoring-rule-89c8222e31c, Jul. 2021.

[12] Tim Roughgarden. CS269I: Incentives in computer science, Lecture 18: Prediction markets, 2016.

Appendices

A Additional results on revenue vs. accuracy

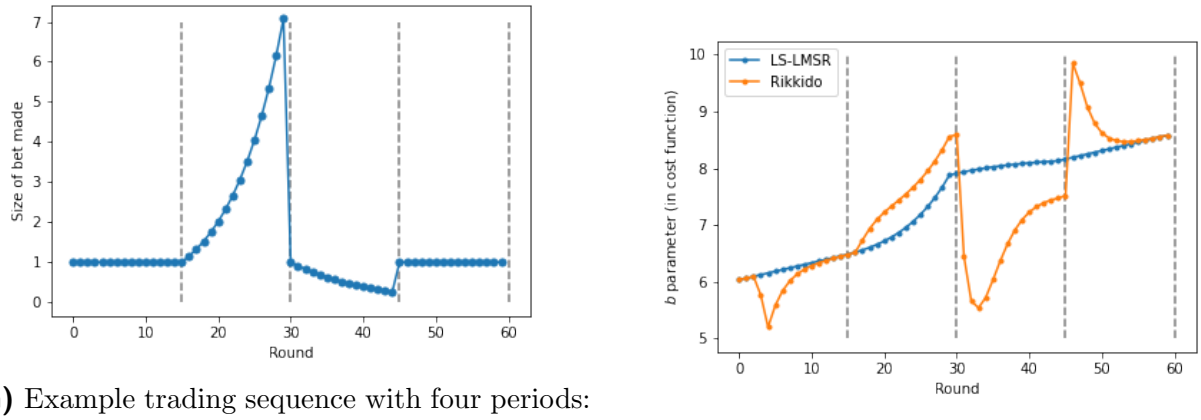


(a) LMSR

(b) LS-LMSR

Figure 5: Realized revenue against price accuracy over 100 trials. Ground truth p drawn from $(0.1, 0.2)$.

B Investigations of Zeitgeist's Rikkido Scoring Rule



(a) Example trading sequence with four periods: (1) constant trades of size 1, (2) exponentially increasing trade sizes, (3) exponentially decreasing trade sizes, (4) back to constant trades of size 1.

(b) Rikkido and LS-LMSR values of the b parameter in the cost function, over the bet sequence presented in Figure 6a.

Figure 6: How Rikkido and LS-LMSR vary over periods of varying volatility (for an example bet sequence).

The key idea of the Rikkido scoring rule is to modify the fixed α parameter of LS-LMSR in such a way that will discourage trading (due to higher prices) during high volatility periods,

while encouraging trading during low volatility periods. Specifically, the fixed α is modified to $\alpha = f + \eta(r)$ [11], where f is again some fixed fee but now r is a term that quantifies the volatility in the market, and η is some non-decreasing function. The initial Zeitgeist PM development of Rikkido defines r as a ratio of *volume* in the market in the last t periods compared to the last $t + k$ period (using an exponential moving average), such that high r means high volume recently compared to before. Thus a larger r contributes to a higher α, which in turn discourages trading because prices increase with α.

To visualize and investigate this behavior numerically, we compared the overall b parameter ($b = \alpha \sum_i q_i$) in LS-LMSR and Rikkido, over the course of a trading sequence with four periods shown in Figure 6a. The corresponding effects on the Rikkido and LS-LMSR b values is shown in Figure 6b. The key takeaway is that while LS-LMSR b is non-decreasing (given that α is fixed and quantity is non-decreasing), the Rikkido b fluctuates more — in particular, increasing more in higher volatility (higher volume) periods, while sharply dropping in value to encourage trading when volatility begins falling.