Athos Vinícius Cavalcante Carvalho

An Agent-Based Framework for Prediction Markets

Brasília

2021

Athos Vinícius Cavalcante Carvalho

An Agent-Based Framework for Prediction Markets

Dissertação apresentada ao Programa de Pós-Graduação em Economia da Universidade de Brasília como requisito para a obtenção do título de Mestre em Economia.

Universidade de Brasília – UnB

Faculdade de Administração, Economia, Contabilidade e Gestão Pública

Programa de Mestrado em Economia

Supervisor: Daniel Oliveira Cajueiro

Brasília

2021

Athos Vinícius Cavalcante Carvalho

An Agent-Based Framework for Prediction Markets

Dissertação apresentada ao Programa de Pós-Graduação em Economia da Universidade de Brasília como requisito para a obtenção do título de Mestre em Economia.

Trabalho aprovado. Brasília, 17 de dezembro de 2021:

Daniel Oliveira Cajueiro
Orientador

Professor
Convidado 1

Professor
Convidado 2

Brasília
2021

Este trabalho é dedicado ao meu pai e minha mãe, por me ensinarem a importância do conhecimento, e a Brasília, por fornecer o cenário.

Acknowledgements

I would like to acknowledge first my advisor, Daniel Oliveira Cajueiro, for providing valuable insights and great understanding during my master's. I am grateful to my family for the support, and my friends for all the good times we spent together.

I also thank Norman Ehrentreich for providing the code used as the basis of this dissertation and Alexandra Elbakyan for working towards a world where science is available to everyone.

Prediction is very difficult,
especially if it's about the future.
(Niels Bohr)

Resumo

Prediction Markets (Mercados Preditivos, em tradução livre) são mercados em que ativos relacionados ao acontecimento de eventos específicos são vendidos. Mercados do tipo são utilizados no mundo empresarial para a previsão da receita de determinado produto ou cumprimento de metas. Esta dissertação apresenta um modelo baseado em agentes, denominado LMSR-ASM, para a avaliação de mercados preditivos. Com a capacidade de testar diferentes tipos de Automated Market Makers (Formadores de Mercado Automáticos, em tradução livre), que são funções matemáticas ou algorítmos computacionais necessários para o fornecimento de liquidez em Mercados Preditivos, o modelo apresentado elucida questões sobre a decisão de parâmetros relacionados à criação de um Mercado Preditivo, assim como a capacidade de geração de lucro desses mercados. É possível simular diferentes tipos de trajetória probabilística de eventos, diferentes formadores de mercado e comportamento dos agentes. Esta dissertação também analisa uma série de questões sobre Mercados Preditivos, demonstrando o impacto da escolha de preços iniciais no lucro do mercado e as oportunidades de receita na implementação computacional dos Formadores de Mercado Automáticos. O trabalho demonstra que o LMSR-ASM pode ser utilizado para encontrar os valores que maximizam lucro para diferentes parâmetros. O modelo também é utilizado para mostrar o comportamento de diferentes Formadores de Mercado Automáticos sob diversas configurações.

Palavras-chave: Mercados Preditivos. Automated Market Makers. Modelos Baseados em Agentes.

Abstract

Prediction Markets (PMs) are exchanges in which agents trade event contingent assets. Many enterprises use PMs as a forecasting tool for a product's revenue and project deadlines. This dissertation presents an Agent-based model, called LMSR-ASM, to evaluate Prediction Markets. Capable of testing different types of Automated Market Makers (AMMs), which are mathematical functions or computational mechanisms needed to provide liquidity in Prediction Markets, the model presents insights into how to set parameters in a PM, as well as how profits react to different settings and AMMs. The model is able to simulate different probability processes, different AMMs and agent behavior. This dissertation also utilizes the LMSR-ASM to evaluate the impact of choosing initial prices in profits and revenue opportunities regarding AMM computational implementation. We show that the LMSR-ASM can be used to find optimal parameters for maximizing profits in PMs and how different AMMs affect market results under a variety of settings.

Keywords: Prediction Markets. Automated Market Makers. Agent-Based Modeling

List of abbreviations and acronyms

CARA	Constant Absolute Risk Aversion
DeFi	Decentralized Finance
LMSR	Logarithmic Market Scoring Rule
LS-LMSR	Liquidity Sensitive Logarithmic Market Scoring Rule
MSR	Market Scoring Rule
PM	Prediction Market
SFI-ASM	Santa-Fe Institute Artificial Stock Market

Contents

1 Introduction

Prediction Markets (PMs), also called Idea Futures or Information Markets (HANSON, 1996), are forums for trading contracts that yield payments based on the outcome of uncertain events (ARROW et al., 2008). A more intuitive explanation for a PM is a market for assets that pay depending on the realization of a certain event. Examples of possible PMs are "Will it rain next month?" or "Who will be elected president in X country?", in which assets are "Yes" and "No" for the former and one for each candidate, for the latter. A binary PM is a market for an event with two possible options. PMs can be created for any event with finite outcomes.

Prediction Markets are extensively used in corporations, regarding the amount of sales of a determined product, future revenue and whether projects will be finished in the proposed timeline, with great success (see Arrow et al. (2008) for examples and Cowgill and Zitzewitz (2015) for an evaluation). There are also sites available to the general public such as PredictIt, the Iowa Electronic Markets and Betfair, with multiple markets, mostly related to politics. There are DeFI[1] apps that make it possible for users to create their own PMs, using the Augur and Gnosis protocols, working on the Ethereum blockchain.

Prediction Markets in general have few participants, which makes liquidity a hindrance to trade and, therefore, to aggregate forecasts. Continuous double auctions (a system that executes trades by matching compatible sell and buy orders in terms of price (DAS et al., 2001)), and order books (a collection of price contingent buy and sell orders (ROşU, 2009)), as used by traditional stock exchanges (PENNOCK, 2004), would only be efficient in a market with a volume of trades most PMs do not have. Likewise, the use of human market makers would raise costs for the operation of PMs. Thus, Prediction Markets need a cost effective way to create liquidity for its assets. This necessity leads to Automated Market Makers (AMMs), which are mathematical functions or computational mechanisms capable of providing liquidity automatically, handling buy and sell orders in a market.

Two AMMs are the focus of this dissertation, the Logarithmic Market Scoring Rule (LMSR), defined in Hanson (2007) and an extension in which one is allowed to vary liquidity, the Liquidity Sensitive Logarithmic Market Scoring Rule (LS-LMSR), defined in Othman et al. (2013). Since the outcomes presented in a PM are exhaustive, prices reflect the relative probability of each result. Therefore, they serve as a proxy for the probability of each outcome (WOLFERS; ZITZEWITZ, 2006). Some AMMs, such as the Dynamic

[1] A Decentralized Finance (DeFi) Application is, as defined by Wang (2020), a smart contract stored in a public distributed ledger (such as a blockchain). It is possible to automate the execution of financial instruments and digital assets using these smart contracts.

Pari-Mutuel Market, presented in Agrawal et al. (2009) do not reflect probabilities in prices. PMs using LMSR can have prices interpreted as probabilities, while the LS-LMSR does not directly do so, but presents a range of possible probabilities related to prices. Slamka, Skiera and Spann (2013) present a review of some AMMs and their characteristics, describing how they vary in parameters, arbitrage conditions and design.

In this dissertation, we design an agent-based computational framework for simulating prediction markets[2], the Logarithmic Market Scoring Rule Artificial Stock Market (LMSR-ASM), using the Santa-Fe Institute Artificial Stock Market (SFI-ASM) as a starting point. Similar works with PMs and agent based models include Slamka, Skiera and Spann (2013), Klingert and Meyer (2018) and Brahma et al. (2012). Our approach differs from theirs due to being the only one evaluating LS-LMSR, as well as adding probability shifts in each period of simulation and offering multiple options for agent behaviors. The LMSR-ASM is capable of simulating a market with two stocks, representing a binary PM, for different AMMs, types of agents and probability trajectories of the underlying event.

To demonstrate the capabilities of the LMSR-ASM, we present the results of simulations regarding the sensitivity of profits in relation to market characteristics, such as the number of participants, as well as choices made by the creator of a prediction market, such as rounding methods, and parameter values. We answer five research questions: how to find optimal parameters for market creation; how profits differ under different AMMs; how rounding floating point calculations affects results; how initial price setting can change market outcomes and how the sum of prices changes in relation to outstanding quantities in LS-LMSR. We replicate expected behavior but also produce novel results. We find that under certain circumstances, the LS-LMSR can be less profitable than the LMSR. We also find that the LMSR-ASM can aid in finding optimal parameters for profit maximizing a PM. The model also shows that different rounding choices can increase profits without hindering market activity. The LMSR-ASM provides a variety of settings to enable further research.

The second chapter introduces essential ideas on Prediction Markets and Automated Market Makers. The third chapter presents the LMSR-ASM and the the Santa Fe Institute Artificial Stock Market, which we used as a basis for our work. Chapter four explains tests applied and their results. The fifth chapter concludes.

[2] The source code for the model can be found at <https://github.com/Athosvcc/LMSR-ASM>. A user guide and class descriptions are presented in the appendix, for further comprehension of the code.

2 Prediction Markets and Automated Market Makers

2.1 Prediction Markets

Prediction Markets encounter difficulties in knowledge aggregation[1] due to low liquidity, which leads to the necessity of a market maker, willing to buy or sell stocks at any time for a determined price. This chapter discusses Automated Market Makers, mathematical functions or computational mechanisms capable of providing liquidity automatically.

2.2 Automated Market Makers (AMMs)

To deal with the low liquidity problem, there is a variety of Automated Market Makers, with the most relevant being the Logarithmic Market Scoring Rule (LMSR). A modification of the LMSR, the Liquidity Sensitive Logarithmic Market Scoring Rule (LS-LMSR) has relevant aspects in relation to their use in Prediction Markets. This chapter presents a discussion of the characteristics and differences of both AMMs. There are other AMMs, for instance, the Dynamic Pari-Mutuel Market (PENNOCK, 2004), and Constant Product AMMs (WANG, 2020), both of which are outside of the scope of this dissertation.

2.2.1 Market Scoring Rules

Scoring rules are functions designed to elicit probability estimates from agents, by rewarding given probabilities in relation to their distance to real outcomes. In a Proper Scoring Rule, as defined by Bickel (2007), an agent maximizes their expected score by revealing their truthful probability expectation. Proper Scoring Rules are efficient in aggregating information regarding probabilities from a single individual. However, joining estimates made by different individuals is not trivial.

Hanson (2003) introduces Market Scoring Rules (MSRs), functions which pool probability estimates from different agents. MSRs work as sequentially shared scoring rules. At any time, agents can decide to interact with the scoring rule by paying the last person who used it. Interpreting a MSR as an automated market maker, this means the last person to interact with the function (buying stock) changes the price of the asset, therefore, in a sense, paying the previous user, which can sell his stocks at the new price.

---

[1] Knowledge aggregation refers to the use of a price system as a mechanism for communicating dispersed information, as shown in Hayek (1945).

A market maker using a MSR in fact only pays the last trader and receives payment from the first trader (CHEN; PENNOCK, 2012).

2.2.1.1 The Logarithmic Market Scoring Rule (LMSR)

The LMSR is a Market Scoring Rule formulated in Hanson (2007), widely used in PMs, due to the property of all prices summing to unity.

The general formula for the LMSR is a cost function of the total of assets in the PM, given by:

$$C(\boldsymbol{q}) = b \log \left(\sum_{j=1}^{n} e^{q_j/b} \right), \tag{2.1}$$

with $\boldsymbol{q}$ being a vector of quantities, b being a strictly positive parameter which controls liquidity in the market, and n being the number of assets (outcomes) in the market. This cost function represents the total amount of money spent on buying or selling all shares available in the market (SLAMKA; SKIERA; SPANN, 2013).

The derivative of the cost function returns the price function for infinitesimal quantities, given by:

$$p_i(\boldsymbol{q}) = \frac{e^{q_i/b}}{\sum_{j=1}^{n} e^{q_i/q}} \tag{2.2}$$

Since a unit of an asset does not usually approach an infinitesimal part of all outstanding quantities, the derivative of the cost function may not be a good approximation for prices. Prices are then given by subtracting cost functions evaluated at two different quantity vectors. As an example, consider a prediction market with a binary outcome, in which there are two stocks, called Y and N. Y is a *Yes* stock, in other words, the stock pays \$1 if the underlying event happens, and N is a *No* stock, which pays \$1 if the underlying event does not happen. In this market, the cost function at an initial quantity vector $\boldsymbol{q}$ is

$$C(\boldsymbol{q}) = b \log \left(e^{\frac{q_{\mathrm{Y}}}{b}} + e^{\frac{q_{\mathrm{N}}}{b}} \right) \tag{2.3}$$

and the price for one *Yes* stock is the difference between cost functions, as follows

$$p_{\mathrm{Y}} = C((q+1)_{\mathrm{Y}}, q_{\mathrm{N}}) - C(q_{\mathrm{Y}}, q_{\mathrm{N}}). \tag{2.4}$$

The creator of a PM sets variable b, which represents the liquidity of the market. The higher b is, the less a fixed value investment moves prices. Market maker's loss is a function of this parameter, with a worst case loss of $b \log n$ (HANSON, 2007). The LMSR is modular, being able to handle stocks with an arbitrary number of outcomes, from binary markets to combinatorial ones, with its only requirement being that assets represent all possible final states. Maximum loss in LMSR is achieved when the market for an event converges to a corner value before market closure. In less extreme cases, market maker loss is given by the difference between the entropies of the initial and final distributions.

Due to consistent losses for the market maker, it is not ideal to use the LMSR in for-profit exchanges. It is possible to avoid losses by imposing costs on transactions, for instance, charging for buy and sell orders or cashing in profits. These costs distort the interpretation of prices as forecasts of the probability of each outcome, since agents factor these fees in their behavior.

2.2.2 A Liquidity Sensitive LMSR (LS-LMSR)

2.2.2.1 Market Maker Properties

Othman et al. (2013) presents three desirable characteristics in a pricing rule:

1. Path independence: the cost of changing the quantity vector from $\boldsymbol{q}^0$ to $\boldsymbol{q}^1$ depends only on $\boldsymbol{q}^0$ and $\boldsymbol{q}^1$, not on the path between them
2. Translation Invariance: prices always sum to unity
3. Liquidity Sensitivity: liquidity changes according to the number of outstanding quantities

Path independence guarantees the market maker cannot become a money pump: any sequence of trades results in the same overall cost, so agents are not able to generate profits only by placing a set of trades. *Translation invariance* makes it so prices can be directly interpreted as probabilities and *liquidity sensitivity* replicates the behavior of stock markets regarding outstanding quantities, the more available stocks a market has, the less a fixed value investment moves prices.

The authors provide proof that a cost function satisfying all three characteristics does not exist. In order to create a liquidity sensitive market maker, one of the other two properties should be abandoned, which leads to the Liquidity Sensitive Logarithmic Market Scoring Rule (LS-LMSR), a market maker presenting path independence and liquidity sensitivity, relaxing the translation invariance property.

Keeping path independence is important to guarantee that, theoretically, the market maker cannot become a money pump. Path independence also ensures what Othman et al. (2013) calls a minimum representation state, meaning any possible market condition can be represented only by the respective quantity vector.

2.2.2.2 Adding Liquidity Sensitivity to the LMSR

In order to add variations in liquidity, the LS-LMSR turns the parameter b of the original LMSR into a function of the outstanding quantities. This way, the original LMSR equation becomes:

$$C(\boldsymbol{q}) = b(\boldsymbol{q}) \log \left(\sum_{j=0}^{n} e^{q_j / b(\boldsymbol{q})} \right), \tag{2.5}$$

where

$$b(\boldsymbol{q}) = \alpha \sum_i q_i \tag{2.6}$$

with α being a strictly positive parameter set by the owner of the PM.

Due to the relaxation of translation invariance, prices in LS-LMSR do not sum to unity. Instead, at any quantity vector, the sum of all prices follows strict lower and upper bounds:

$$1 \leq \sum_i p_i(\boldsymbol{q}) \leq 1 + \alpha n \log n \tag{2.7}$$

The sum of all prices approaches the lower bound when $q_i \to \infty$ and $q_j = 0$ for $i \neq j$. The upper bound is reached when all quantities are equal, $\boldsymbol{q} = k_1$ for any k_1.

Othman et al. (2013) shows relaxing translation invariance makes it so prices in LS-LMSR do not directly translate to probabilities. When the sum of prices reaches its upper bound, any probability between $\frac{1}{n} - \alpha(n-1)\log n$ and $\frac{1}{n} + \alpha(n-1)\log n$ would be consistent with current prices.

2.2.2.3 Choosing the parameter α

The choice for the parameter α can follow an intuition from traditional betting markets. Since the sum of prices is dependent on α, the creator of the market can set an upper bound for its possible maximum commission, also called vigorish, or the *vig*. For any desired *vig* v, we can guarantee that maximum commission does not exceed v by setting the parameter α as follows:

$$\alpha = \frac{v}{n \log n}. \tag{2.8}$$

As an example, for a maximum vig of 20%, common in bookmaker markets (SMITH; PATON; WILLIAMS, 2006), and a PM with two possible outcomes, the parameter α should be set to

$$\alpha = \frac{0.2}{2 \log 2} \approx 0.1442.$$

2.2.2.4 Market Maker Revenue and Loss

Like the original LMSR, the LS-LMSR exhibits bounded loss, but instead of $n \log n$, it is given by the cost function at initial quantities $C(\boldsymbol{q^0})$. This means the market maker can lower their losses by setting initial quantities as low as is desired. This has a trade-off in initial market liquidity. In a market with $\alpha = 0.05$ and with only one outstanding quantity, an initial bet would cost over \$0.95 (OTHMAN et al., 2013), so even though it is possible to make worst case loss approach zero, making it so might discourage trading.

Regarding revenue, for markets with the same worst case loss, the LS-LMSR generates equal or more revenue than the LMSR for the same set of trades, no matter the outcome or outstanding quantities. Worst case revenue can be presented in closed form and is given by the equation

$$\mathcal{R}(\boldsymbol{q}) \equiv C(\boldsymbol{q}) - \max_i q_i - C(\boldsymbol{q}^{\mathbf{0}}). \tag{2.9}$$

Thus, the appeal of the LS-LMSR also comes from achieving profits more frequently in relation to the original LMSR, in exchange for a marginal loss in the interpretation of prices as probabilities, being a better choice when running for-profit Prediction Markets.

3 The LMSR-ASM Agent-Based Model

This chapter presents the Santa-Fe Institute Artificial Stock Market (SFI-ASM), which is used as a basis for the LMSR-ASM. We then relate the SFI-ASM with our model.

3.1 The Santa-Fe Institute Artificial Stock Market

The Santa-Fe Institute Artificial Stock Market (SFI-ASM) is an agent-based model for stock market simulations, originally described in the 1989 paper *"Artificial economic life: a simple model of a stock market"* (PALMER et al., 1994). In Lebaron (2002), Blake D. LeBaron, one of the model's creators, presents a history of the design of the SFI-ASM, as well as a review of early stock market agent-based models. Johnson (2005) presents a repository with information on different versions of the SFI-ASM. In the website, there are versions made in Objective-C (by Brandon Weber and Paul Johnson) and Java (by José Manuel Galán and Luis R. Izquierdo), using the Swarm toolkit.

For Prediction Market simulations, we adapted the version created by Ehrentreich (2008), developed in Java using the Repast library.

3.1.1 Basic Structure

In Norman Ehrentreich's version of the SFI-ASM, there is an arbitrary number of traders, chosen at the beginning of the simulation, each holding one unit of risky stock and 20,000 units of cash. At each period, traders make investment decisions, choosing how much to invest in stocks and how much to keep in cash, which yields a risk-free rate of return r_f.

Agents are myopic, meaning they only consider prices for the next period. They are also homogeneous with respect to their utility function, determining their optimal stock holdings at each period by maximizing a constant absolute risk aversion (CARA) utility, in the form:

$$U(W_{i,t+1}) = -e^{-\lambda W_{i,t+1}}, \tag{3.1}$$

where λ is a strictly positive parameter, representing the degree of risk aversion, and $W_{i,t+1}$ is the expected wealth of agent i for the next period. Agents face the following budget constraint:

$$W_{i,t+1} = x_{i,t}(p_{t+1} + d_{t+1}) + (1 + r_f)(W_{i,t} - p_t x_{i,t}), \tag{3.2}$$

where p_t is price at time t, $x_{i,t}$ is the amount of stock held by agent i at period t, and d_{t+1} is a stochastic dividend which the stock pays, given by a mean-reverting autoregressive

Ornstein-Uhlenbeck process, of the form

$$d_{t+1} = \overline{d} + \rho(d_t - \overline{d}) + \epsilon_{t+1}, \tag{3.3}$$

where $\overline{d}$ is the dividend mean, ρ is the speed of mean reversion and ϵ are stochastic shocks following $\mathcal{N}(0, \sigma^2{}_\epsilon)$.

The solution to the utility maximization returns the optimal risky stock holding at any period, which is given by the equation

$$\widehat{x_{i,t}} = \frac{E_{i,t}[p_{t+1} + d_{t+1}] - p_t(1 + r_f)}{\lambda\sigma^2_{t,p+d}}, \tag{3.4}$$

with $\sigma^2_{t,p+d}$ being the empirically observed variance of the stock's price plus dividend time series, $E_{i,t}[p_{t+1} + d_{t+1}]$ being agent i's expectation of next period's stock price and dividend.

Traders derive their expectations by a set of trading rules, which are conditions unique to each agent, with each condition based either in fundamental or technical analysis of the dividend process. Example of a fundamental rule would be "dividend–price ratio is under $\frac{1}{2}$" and an example of a technical rule would be "the 25-period moving average of the stock price is greater than the current price". According to these Boolean conditions, together with fitness values for each rule, agents make their forecasts. Different expectations represent the way different agents process the same information set available to all traders. In the SFI-ASM, agents can learn, changing their trading rules by feedback learning, observing which rules produce better forecasts over time, and a genetic algorithm, which enables agents to change rules via mutation and cross-over processes.

3.2 The LMSR-ASM

The LMSR-ASM is an adaptation for Prediction Market simulations of Norman Ehrentreich's Java version of the SFI-ASM (NESFI-ASM), as presented in Ehrentreich (2008). The model utilizes RepastJ 1.4, a Java library for creating agent-based models.

As stated in Section 3.1.1, simulations start with a previously selected number of agents, carrying a predetermined quantity of cash. At each period, agents update their optimal demand for stocks by maximizing a constant absolute risk aversion (CARA) utility. The agents buy or sell stocks according to the difference between their optimal demand, resulting from the maximization problem, and the number of stocks they currently have.

Event probability is either arbitrarily chosen at the beginning of a simulation, or defined by one of the generative processes available, which will be explained in Section 3.2.1. Each agent has a forecast of the probability of the underlying event. This forecast is based on the probability of the event, given by the simulation, plus a random element unique to each agent, following $\mathcal{N}(0, 0.05)$, as defined in Slamka, Skiera and Spann (2013).

This variation models the fact that agents form beliefs according to noisy signals, particular to each one. These signals follow a Gaussian distribution, as presented in Das (2005).

A simulation run has the following workflow: either LMSR or LS-LMSR is selected as the AMM and, in the first period, initial quantities are created as arbitrarily set, either to create liquidity, when in LS-LMSR, or to adjust initial prices, in both cases. These initial quantities are *artificial* stocks. The market maker holds these stocks and, since they're not paid at the end of the simulation, they do not affect profits. The LS-LMSR presents liquidity problems without initial quantities, with the function being undefined for $\boldsymbol{q} = (0, 0)$ and one-unit bet prices being costly at the point of discouraging trades (see Othman et al. (2013)), making these *artificial* stocks necessary to kickstart the market. Regarding price adjustment, initial stocks are created until the point when first-unit bets reach the chosen value. Stock creation is a practical way to affect prices without changing the behavior of cost functions.

After this first period, agents can trade stocks according to their own personal forecasts and, at the last period, the underlying event is performed: the Automated Market Maker closes the market paying the agents with winning stocks according to their holdings. The following section explains stock behavior and how agents make their decisions. Figure 1 summarizes the how the model works.

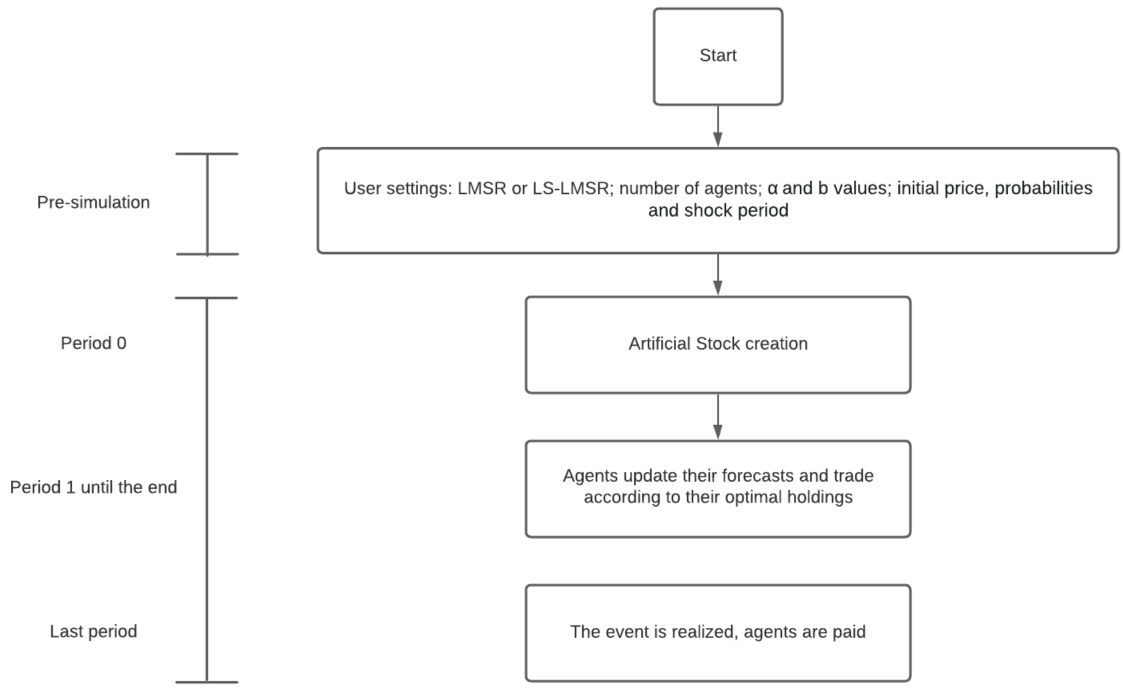

Figure 1 – Flowchart of the LMSR-ASM.

3.2.1 Stock Characteristics and Behavior

There are two types of stocks, one paying \$1 if the underlying event happens, which we will refer to as *Yes* stocks, and another, which pays \$1 if the underlying event does not happen, which we will refer to as *No* stocks.

Both stocks are related to an event, with probability set at the beginning of the simulation. The market maker can also set an initial price for the stocks, achieved by the creation of *artificial* stocks, as explained before. Besides probability, stocks have a liquidity parameter, b for LMSR or α for LS-LMSR.

Stock prices in LMSR sum to 1, while in LS-LMSR, the sum of the two stocks vary according to outstanding quantities, with an upper limit of $1 + \alpha n \log n$, as presented in Chapter 2. Since the event simulated only has two possible outcomes, this upper limit is $1 + \alpha \times 2 \times \log 2$.

It is also possible to include a probability shock at an arbitrary period, in which the probability of the event changes to a preset value. During the rest of the simulation, stock probability can follow three types of movement chosen at will:

- Fixed: the underlying probability does not change unless a shock is specified
- Random Walk: at every step, the event's probability changes in relation to a random shock following $\mathcal{N}(0, 0.05)$
- Logit: the probability of the event is updated following a Logit generative process, explained later in this section

Figure 2 illustrates how each probability setting behaves in a 100 period run. The simulation starts at 0.7 and drops to 0.3, after a shock at period 50.

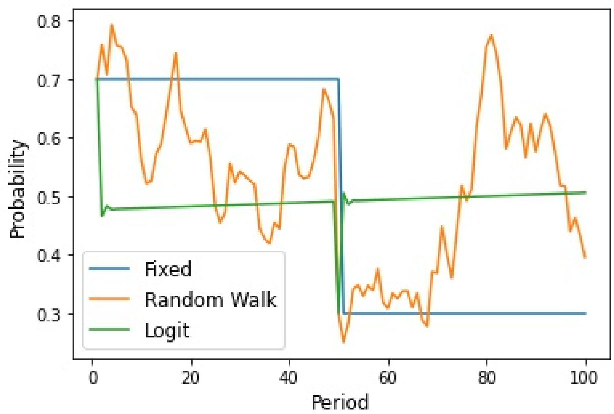

Figure 2 – A demonstration of how different probability processes vary over time.

The shock represents a situation in which probabilities change drastically, for example, the probability of a football team winning a two-game series after losing the first game. The Fixed setting is useful to model the behavior of a stable event, e.g., the probability of a candidate being elected, considering a week with no important news regarding the race. Random walk and Logit options are able to simulate more complex processes if desired.

At the final period, the market closes according to the following rule: if *probability* > 0.5, *Yes* stocks pay out; otherwise, *No* stocks pay out. Therefore, stock behavior is defined as follows:

- At the start of the simulation, quantities are created to set initial prices and liquidity
- Exchanges follow either LMSR or LS-LMSR, as predetermined, with parameters b or α
- Probability is updated at every step according to the process chosen
- At a set period, the underlying event's probability changes from its previous value to another predetermined one
- At the end, the underlying event is performed and agents receive their payments

The logit generated process is given by the equation

$$\log\left(\frac{\pi}{1-\pi}\right) = \boldsymbol{\beta}_1'\boldsymbol{z}_{(t-1)}, \tag{3.5}$$

as described in Fokianos and Kedem (2003), where $\boldsymbol{\beta}_1'$ is a vector of time-invariant parameters and $\boldsymbol{z}_{(t-1)}$ is a covariate process.

Since it is useful for the process to be defined in terms of characteristics in the simulation, the vector $\boldsymbol{z_{t-1}}$ is given as $\boldsymbol{z}_{(t-1)} = (t, \pi_{t-1}, \pi_{t-2})$. This makes the generated sequence highly persistent, but betas and the covariate process can be changed in the source code if an arbitrary sequence of probabilities is desired.

To supply the model with a vector of parameters related to a real event, we ran a beta regression on data from polls regarding Brazil's 2018 election, taken from Poder360 (2021). The betas used are $\boldsymbol{\beta_1'} = (0.02044511, -1.21038522, -1.50537449)$ and the independent variables are as defined by the vector $\boldsymbol{z_{t-1}}$. Figure 3 shows poll data, the predicted fit of the model and the generative process created using the betas given.

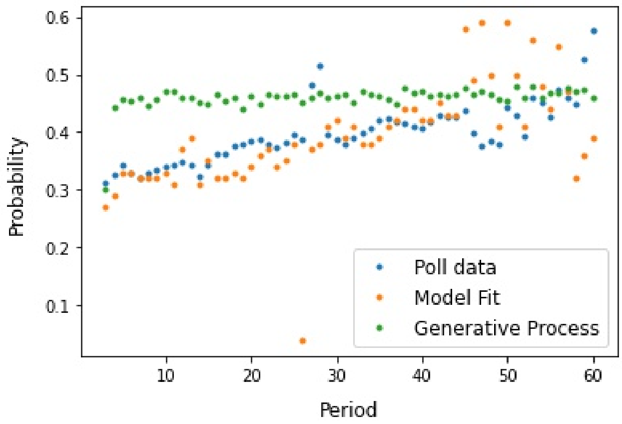

Figure 3 – Poll data, Model Fit and Generated Data from Beta Regression.

Each blue dot represents the results of a different poll, with higher periods meaning polls made later in time. The probability axis shows the surveyed percentage for the candidate with the highest vote intention, with the complementary event being the election of any other candidate. Orange dots are the model's fit, using values from the original table and estimated betas from the regression. Green dots are values generated by using the process shown in Equation (3.5), along with a random generator for the data, which follows $\mathcal{N}(0, 0.1)$. As explained before, since the independent variables used were only the period and lagged values, the model's fit and generated data are highly persistent. This effect can also be observed in Figure 2, where the logit generated probabilities do not change greatly after a 1-period probability shock. If this characteristic is unwanted, other covariate processes might be better suited for simulation of probability processes.

3.2.2 Agent Characteristics and Behavior

The model also has three distinct types of agents:

- Ideal: a perfect foresight agent, used as a baseline. It has the capability of knowing the probability of the underlying event and buys stock until the price matches the probability
- Logit: the agent creates forecasts according to a unique logit model of the probability process
- Random Walk: the agent creates forecasts according to noisy signals they receive regarding the event, given by the real probability adjusted by a random variable unique to each agent, which follows $\mathcal{N}(0, 0.05)$

Logit and Random Walk agents are utility maximizing, buying and selling stocks only until their current stock holding equals their optimal holdings.

Following the SFI-ASM specification in Section 3.1.1, agents are myopic, meaning they only consider prices for the next period, and homogeneous with respect to their utility function, determining their optimal stock holdings at each period by maximizing a constant absolute risk aversion (CARA) utility, shown in Eq. (3.1).

Due to Prediction Market stocks not paying dividends, the budget constraint of the agents differs from what we see in Eq. (3.2), now being as follows:

$$W_{i,t+1} = p_{t+1}x_{i,t} + (1+r_f)(W_{i,t} - p_t x_{i,t}), \tag{3.6}$$

where p_t is price at time t, $x_{i,t}$ is the amount of stock held by agent i at period t, and r_f is the risk free rate of return. By solving the utility maximization problem, we find the optimal risky stock holding at any period, which is given by the equation:

$$\widehat{x_{i,t}} = \frac{E_{i,t}[p_{t+1}] - p_t(1+r_f)}{\lambda\sigma^2_{t,p}}, \tag{3.7}$$

with $\sigma^2_{t,p}$ being the variance of the stock, which follows a Bernoulli distribution and $E_{i,t}[p_{t+1}]$ being agent i's forecast[1].

Since optimal stock holdings are dependent only on the difference between their forecast and the current price, both of which have an upper bound of 1, due to being related to probabilities, agent's demands rarely exceed 2. In the edge case, with price 0 and an agents forecast of 1, the optimal stock holdings for an agent with $\lambda = 0.3$ would be $\frac{1-0}{0.3\times 1} = 3.\overline{3}$. This does not seem to affect convergence, even though it can make prices volatile in a simulation with a small number of agents, due to few outstanding stocks.

If the agent's forecast of the probability of the underlying event is higher than the current stock price and it does not hold *No* stocks, it buys *Yes* stocks according to its optimal demand. If the agent's forecast of the probability of the underlying event is lower than the current stock price and it does not hold *Yes* stocks, it buys *No* stocks according to its optimal demand. Even though prices in the LS-LMSR do not sum to unity, since the payoff is still \$1, each agent's expected return remains unchanged. Therefore, changing AMMs does not change how agents make choices. In case the agent's prediction is not in agreement with its stock holdings, the agent sells the stocks it currently holds until prices change to match its forecast or it holds no stocks. The following table summarizes agent behavior.

Table 1 – Agent Behavior in the LMSR-ASM

	forecast > price	*forecast < price*
holds 'Yes' stocks	buys 'Yes' according to optimal demand	sells all 'Yes' stocks
holds 'No' stocks	sells all 'No' stocks	buys 'No' according to optimal demand

[1] Since prices can be interpreted as probabilities, the term represents the agent's probability forecast as well as its expectations of the price for the next period

3.3 Automated Market Maker

Users can select either LMSR or LS-LMSR, and choose each AMMs parameters, b or α. At each period, agents are activated one at a time, determine their demands as shown previously and then interact with the market maker. Agents can only buy integer stocks, which many times lead to the necessity of rounding prices. Rounding is also necessary due to floating point imprecision. If the float cost was used directly, it could be possible for the market maker to be used as a money pump. Using floating point values is also a possibility in the model, even though it would not be advisable in a real setting.

In the model, rounding can be done in two ways: either the cost of an order is approximated to the nearest 2 or, as used in Berg and Proebsting (2009), the nearest 6 decimal digits. The market described by Berg and Proebsting has a distinct transaction interface, with agents investing determined values and in return receiving a rounded number of stocks. In my model, agents order the number of stocks they demand and pay accordingly. This means that when rounding prices to the nearest 6 decimal digits, agents many times pay fractions of a cent. In a real setting this could be handled by showing users balances rounded down to the nearest cent but using the actual values of 6 decimal digits during transactions.

The AMM handles selling and buying differently, rounding prices down when agents are selling and up when they're buying. This can be seen either as a form of "breakage" or as a bid-ask spread in the AMM. The rounding behavior helps to offset part of the market maker's loss. With breakage, there is a change in the path independence characteristic of LMSR. While ideally an agent could buy and immediately resell their shares without incurring losses, in the model this action possibly leads to small losses.

3.4 World settings and Repast

Additional settings that can be chosen in the simulation are the risk-free rate of return and the number of periods for simulation. A deeper explanation of the Repast interface can be found in the appendix, as well as other important information about the source code.

4 Results

This chapter presents the results of a series of simulations using the LMSR-ASM. We run simulations in order to answer five research questions:

- What are the ideal values for α and b?
- What are the profit differences between LMSR and LS-LMSR market makers?
- How much do rounding types affect profits?
- How much do initial prices affect profits?
- How the sum of prices vary according to outstanding quantities in LS-LMSR?

Each setting is used in 100 runs, for simulations involving stochastic factors. This value is used so simulations do not take a long time, while simultaneously producing credible results. Outcomes do not vary greatly with a higher number of runs and the analysis remains the same. Only one run for each setting is executed when simulations are deterministic, as is the case with Ideal agents.

Over all following simulations, unless otherwise stated, values set in the LMSR-ASM are:

- Number of periods = 100
- Probability process is Fixed
- $\alpha = 0.04$ and $b = 76.13$
- Initial price = 0.5
- Initial quantities = $(50, 50)$
- Initial probability = 0.8
- Probability shock in period 50
- Probability after shock = 0.2
- Interest rate = 0.0
- Risk aversion = 0.3

The fixed probability process shows how AMMs and agents behave in an ideal market, so it is possible to analyze market characteristics independently of random shocks. The same rationale applies to the interest rate: since the analysis is on PM trading and AMMs, not portfolio decisions, a zero interest rate focuses simulations to the desired behavior.

The planned shock at period 50 enables understanding of a market's reaction in relation to a sudden change in probabilities, after values have already stabilized regarding previous knowledge. Probability values are distant to each other and the initial price to highlight the movement of price convergence in relation to probability. Risk aversion is as set in Ehrentreich (2008), the Santa-Fe Institute Artificial Stock Market used as basis for the LMSR-ASM.

Values for $b = 76.13$ and $\alpha = 0.04$ are equivalent in respect to losses, as they lead to equal worst case loss of ≈ 52.77. As shown in Section 4.1, we use $\alpha = 0.04$ since it is the value that maximizes profits under the parameters given. The maximum *vig* possible is $0.04 \times 2 \times \log 2 \approx 0.0554$, $\approx 5.54\%$ of the market's revenue. Batch files and simulation data can be provided upon request.

4.1 Choosing α and b

To analyze optimal values for the AMMs parameters α and b, we run simulations varying both parameters, as well as the number of participants in a market. The parameter b varies from 15 to 150, in increments of 15; the parameter α varies from 0.01 to 0.10, in 0.01 increments. This variation is used to compare how volume of trades and market profit vary in a range of values. A comparison between runs using different settings enables understanding of how both the LMSR and LS-LMSR behave when their parameters change.

The baseline, a LMSR simulation with ideal agents, returns an increase in the volume of trades in relation to an increase in b, while market profit decreases with the parameter. The results are invariant in relation to the number of participants because, with ideal agents, the first agent is the only one who engages in trading. They are also invariant to the number of periods, since convergence happens few periods after the shock and does not change over time. Volume increases with b because, due to the increase in liquidity, more trades are necessary to move prices to their equilibrium value. Market profit decreases with b because LMSR worst case loss is given by $b \log n$.

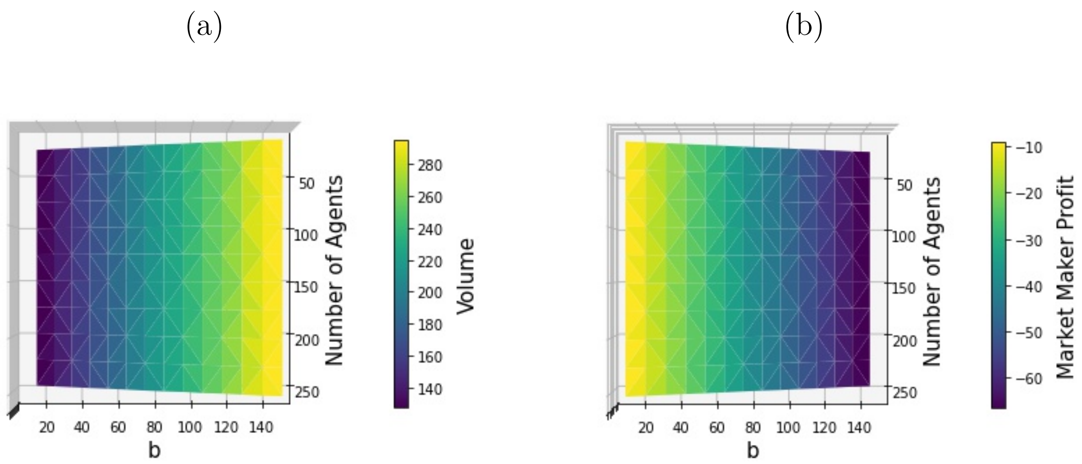

Figure 4 – Parameter analysis in the LMSR model with Ideal agents. (a) Effect in trading volume; (b) Effect in market profit.

Using random walk agents, results for the LMSR are more interesting. Figure 6 shows how liquidity and volume interact with a varying number of agents. Results are as expected from Equation (2.1). Volume of trades increases both with b and the number of agents, while market maker profit is decreasing in respect to b and increasing in the number of participants. There is no optimal rule for choosing the parameter b: it is always possible to increase profits by decreasing the parameter, but it might reduce liquidity so much that prices are not indicative of underlying probability. Figure 5 shows a case where prices do not converge to the probability due to low liquidity. Due to agents updating forecasts every period following a Normal distribution with mean 0, simulating for longer periods than 100 would not change the lack of convergence, prices just continue varying around the true probability value, as shown in 5. Since buying a small number of stocks cause big price increases, we see skips in price in relation to the changing perception of the agents. Prices vary around the probability but do not converge and are highly volatile. Either way, batch simulations can help market creators set their initial parameters, according to desired volumes and profits.

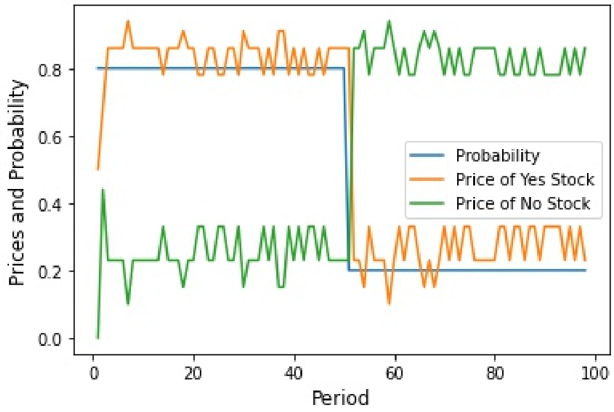

Figure 5 – A run showing lack of convergence to probability when liquidity is low. $b = 2$ is used.

When the created market does not focus on profit, such as an internal PM created by a company, knowledge aggregation is the most important factor in choosing a market's settings. This might mean that trading volume is the primary variable, to account for small differences in agents' perception of the underlying event's probability. Play-money markets can offset a higher liquidity by distributing more tokens to participants, thus, being more efficient in relation to knowledge aggregation, with no drawbacks. For-profit markets deal with the trade-off between volume, related to the liquidity parameter b, and market profit. Especially when considering the LMSR, which rarely is profitable without any additional fees, lower liquidities can completely hinder trade, since agents factor extra costs into their expectations, distorting market prices and lowering overall participation.

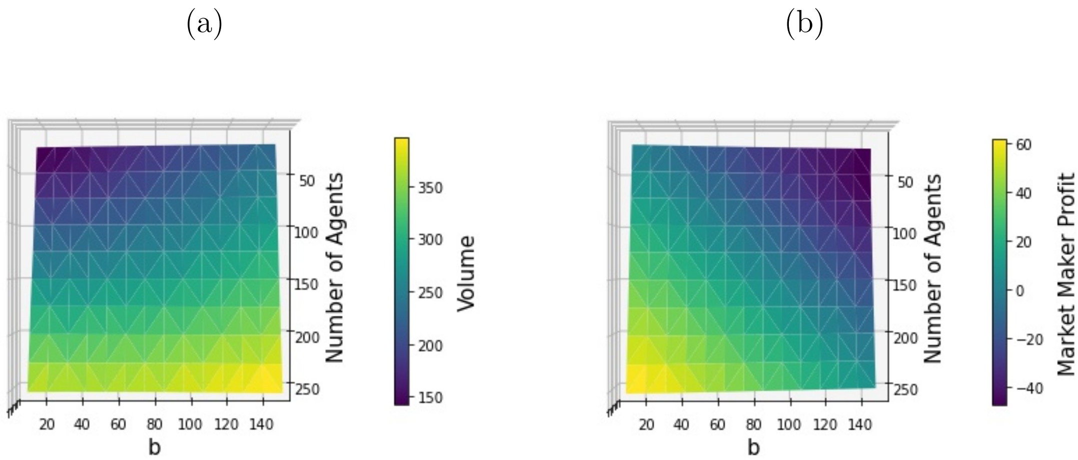

Figure 6 – Parameter analysis in the LMSR model with Random Walk agents. (a) Effect in trading volume; (b) Effect in market profit.

Regarding the LS-LMSR, since liquidity and profits are directly interlinked by the parameter α, there is an optimal choice for each market, a value which maximizes market maker profit. Lower α values decrease worst case loss, but also decrease liquidity to the point of reducing volume and market values, which in turn affect profits. However, higher α values also decrease volume. Agents trade less because prices are not directly translated to probabilities. As presented in the previous chapter, when the sum of prices reaches its upper bound, given by $1 + \alpha n \log n$, any probability between $\frac{1}{n} - \alpha(n-1)\log n$ and $\frac{1}{n} + \alpha(n-1)\log n$ would be consistent with stock prices. Figure 7 shows volume and market maker profit for the simulated parameters, with Random Walk agents.

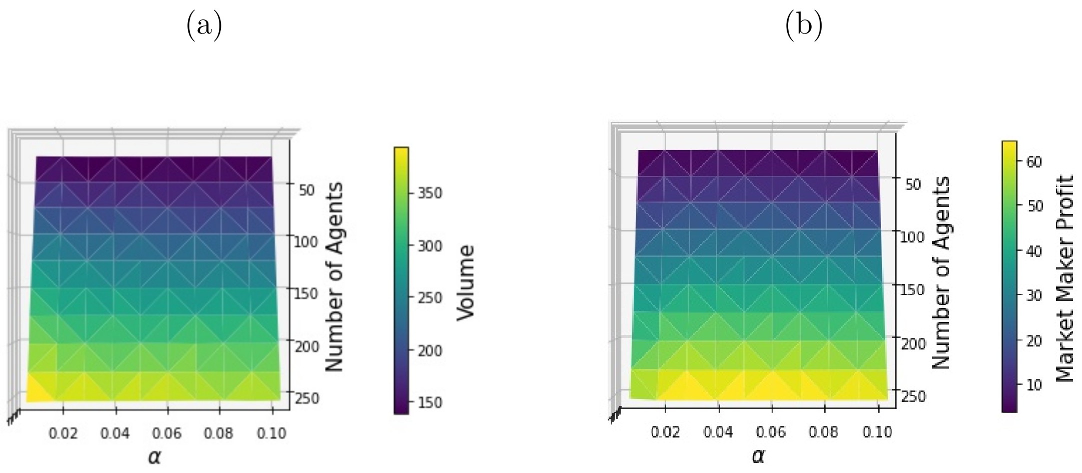

Figure 7 – Parameter analysis in the LS-LMSR model with Random Walk agents. (a) Effect in trading volume; (b) Effect in market profit.

Volumes increase with respect to the number of agents, and vary slightly regarding α values, mostly due to different seeds in the Random Walk behavior of the agents. The profits graph (Figure 7(b)) presents the behavior described previously. Market maker profit increases with α until it reaches its maximum value, at $\alpha = 0.04$. This value may not be consistent with different types of agent or probability behavior, but it is useful for further simulations and also to demonstrate the model's capacity to find an optimal value under a determined setting. The possibility of finding an optimal value for the parameter α using simulated data is novel and enables PM creators to maximize profits by considering expected market characteristics.

4.2 LMSR vs LS-LMSR

To evaluate the differences between LMSR and LS-LMSR, we vary the probability after shock as well as the underlying probability, from 0.1 to 0.9, inclusive. This range shows how the AMMs react under diverse situations, for the spectrum of possibilities. We use ideal agents for baseline, and Random Walk agents to simulate a market with more uncertainty.

Figure 8 shows the results:

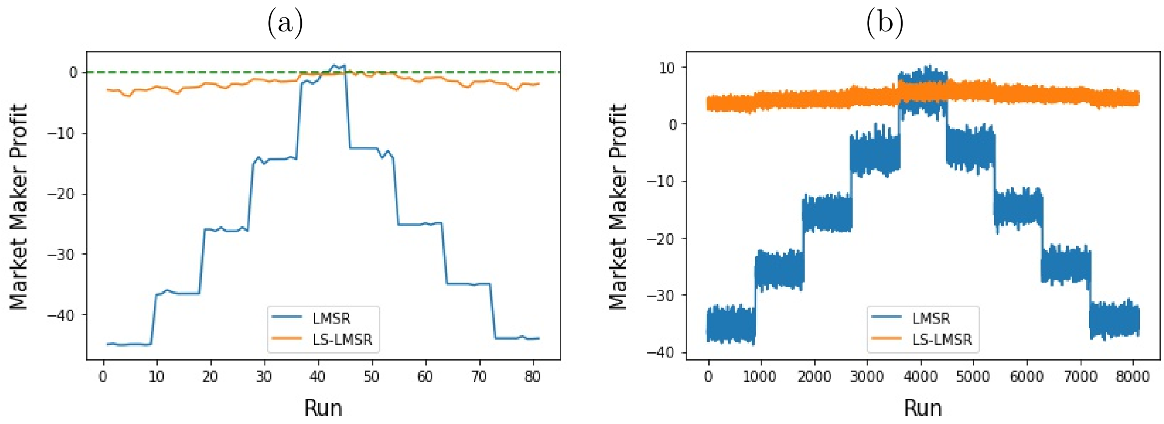

Figure 8 – Comparison between LS-LMSR and LMSR AMMs. (a) Using Ideal agents; (b) Using Random Walk agents. The dotted green line separates profits and losses.

Figure 8 shows that the LS-LMSR consistently presents higher profits than the LMSR. However, in certain cases, LMSR and LS-LMSR profits converge. This happens when after shock probabilities are similar to both initial probabilities and prices (the middle of the graph). Due to the set of trades not being equal between simulations, in some runs, the LMSR is more profitable than the LS-LMSR. It is possible for the LMSR to be more profitable than the LS-LMSR when under these edge situations, due to changes in behavior caused by the different liquidities. LS-LMSR revenue is always higher for an equal set of trades, but since agent incentives are not the same under different AMMs, trades are also not equal. For less edge cases, LS-LMSR not only presents higher profits, but also reduces variance, increasing the predictability of a market's result.

Simulations with Random Walk agents present higher profits than with Ideal agents because the market maker revenue increases with the number of trades, which are higher in Random Walk simulations due to agents constantly updating their forecasts. As explained in the beginning of the chapter, for simulations involving stochastic factors, each setting is used in 100 runs, while only one run for each setting is executed when simulations are deterministic, as is the case with Ideal agents. Therefore, the Random Walk simulation has 100 times more runs than the one with Ideal agents. Since the same settings are used, the simulations are directly comparable.

4.3 Rounding and Profits

Due to floating point imprecision, cost calculations need to be adapted so the market maker will not become a money pump (BERG; PROEBSTING, 2009). The way a PM creator decides to make these adaptations can reflect on PM revenue. We evaluate two different rounding methods, rounding costs up by the nearest of two decimal digits or by the nearest of six decimal digits, as used in Berg and Proebsting (2009). Even though this difference in rounding does not affect behavior significantly, it has a discernible effect in profits, especially when the amount of trading is large.

To test rounding effects in PM profit, we run the model under the same settings as the previous test, with the probability after shock and underlying probabilities changing from 0.1 to 0.9. Figure 9 shows the effects:

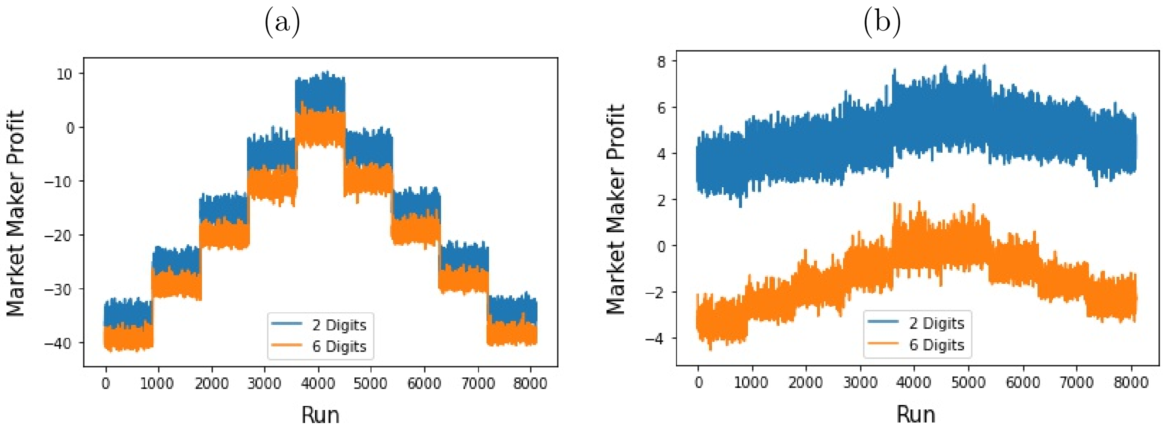

Figure 9 – Evaluation of rounding methods: 2-digit rounding vs 6-digit rounding. (a) LMSR; (b) LS-LMSR.

As expected, rounding to 2 decimal digits increases profits in both LMSR and LS-LMSR. The LMSR varies more in general in relation to initial conditions, as shown in Section 4.2. The LS-LMSR presents significantly less variance in profits in relation to initial settings. Either way, rounding to 2 decimal digits presents a way to increase profits that does not affect market participation greatly.

4.4 How important are initial prices

As presented in Hanson (2007), the expected cost of running a market is minimized by setting the initial report (price) equal to the market creator's beliefs. To evaluate the impact of price setting, we run the model with a fixed probability, while varying initial prices, as well as the probability after shock. The batch run used values of 0.1 to 0.9 for both variables, using Ideal and Random Walk agents. Figure 10 shows the results:

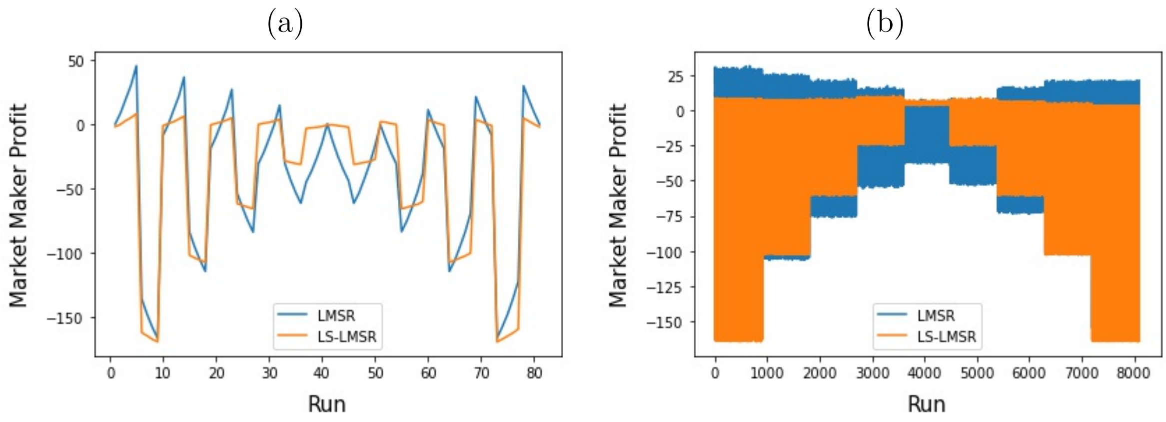

Figure 10 – How initial price setting affects profits. (a) With Ideal agents; (b) With Random Walk agents.

The simulation shows that the LS-LMSR only presents higher profits than the LMSR when prices are set to 0.5, which are the results in the middle of the graph. Positive spikes are simulations in which the market maker sets initial prices to the correct probability and negative spikes are simulations in which the opposite occurs. Simulations with Random Walk agents confirm the phenomenon: when initial prices approach corner values, maximum loss in LS-LMSR is greater than in the LMSR, with the opposite happening when initial prices approach equality. This can be useful in PMs in which the market creator has strong priors regarding the final probability. In these cases, using LMSR and setting initial prices accordingly might increase profits. It is also of note that setting initial prices introduces greater variance in profits. The possibility of the LMSR being more profitable then the LS-LMSR when prices are not equal at the beginning is unexpected and a novel result, showing how the LMSR-ASM can extend our understanding of AMM characteristics.

4.5 Sum of Prices and Quantities in LS-LMSR

The sum of prices in LS-LMSR reaches its maximum when the quantity of assets are equal and approaches unity as the quantity of one of the stocks goes to infinity. Figure 11 demonstrates how the sum changes as the ratio between stocks changes, for three different values of the parameter α, the upper and lower bounds used in previous simulations, and the profit maximizing value of $\alpha = 0.04$:

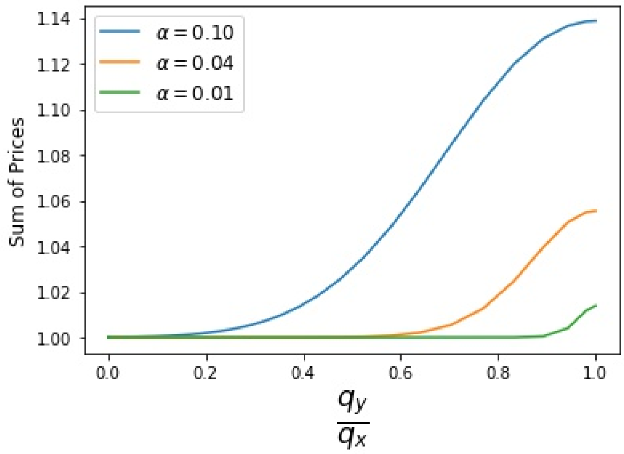

Figure 11 – Changes in the sum of prices in LS-LMSR, according to the ratio between stocks, for different values of α.

The plotted values show another characteristic of the LS-LMSR and its sensitivity regarding the choice of the parameter α: not only the parameter defines the maximum profit possible, it also sets how relative quantities affect profitability. While a market with $\alpha = 0.04$ and a 2:1 ratio has almost converged to the lower bound of profitability, the same quantities still lead to a vig of approximately 4% with $\alpha = 0.10$. This leads to a rule of thumb in market creation: when convergence to a corner price is expected, a higher alpha is needed to generate consistent profits. Analysis of the underlying events are important in market creation.

5 Conclusion

Prediction Markets are efficient in aggregating information regarding the likelihood of uncertain events. Due to liquidity problems, most PMs utilize Automated Market Makers to guarantee agents can buy and sell shares at any moment in time. A variety of AMMs exist, with different cost functions and underlying behavior, which in turn affect agent participation. The most studied mechanism is the Logarithmic Market Scoring Rule (LMSR). The LMSR can be altered to be liquidity sensitive (LS-LMSR).

We adapted the Santa-Fe Institute Artificial Stock Market to run Prediction Market simulations under different initial settings in order to address how agent behavior and market structure change profits for different types of AMMs and their respective settings.

The model, called LMSR-ASM, demonstrates it is possible to set an optimal parameter value for profit maximization when using the LS-LMSR, while LMSR parameters depend on subjective evaluation of a market's creator, due to a trade-off between profits and market volume. Other results are that market maker loss is sensitive to initial prices, making it worthwhile for owners of PMs to start markets with prices as close as the true probability in case underlying events do not converge before realization. For some cases, setting initial prices might increase profit variance and make the LS-LMSR less profitable than the LMSR. We also address different calculation rounding options, showing that a necessary solution to floating point imprecision can be used to increase profits without affecting market behavior greatly.

The LMSR-ASM reproduces the expected behavior, showing how profits decrease for higher values of the parameter b in LMSR and how the LS-LMSR is more profitable under certain circumstances. In addition, the model also produces novel results, such as showing the possibility of parameter optimization in LS-LMSR, and how the LS-LMSR reduces variance in profits when initial prices are \$0.5, but can actually be less profitable, for a range of other initial prices.

When using the LS-LMSR, a market creator can set liquidity in a way such that profits are more likely, but ensuring profits can hinder the PMs capacity of aggregating information, since transaction costs change agent behavior, leading to less convergence. In play money markets losses are not undesirable, so liquidity should be set low enough that agents can influence prices but at the same time not so low that prices change greatly with small orders. This makes it so PM managers should evaluate parameters in a case by case basis. The LS-LMSR has the highest capability of generating profits, when initial prices are set to 0.50. On this setting, without any additional market fees, the LMSR generally leads to losses for the market maker.

The source code can be found at <https://github.com/Athosvcc/LMSR-ASM> and can be easily adapted for different types of agents, market makers, and events.

Bibliography

AGRAWAL, S. et al. A Unified Framework for Dynamic Pari-Mutuel Information Market Design. *arXiv:0902.2429 [q-fin]*, feb. 2009. ArXiv: 0902.2429. Available at: <http://arxiv.org/abs/0902.2429>. Cited in page 12.

ARROW, K. J. et al. The Promise of Prediction Markets. *Science*, v. 320, n. 5878, p. 877–878, may 2008. ISSN 0036-8075, 1095-9203. Publisher: American Association for the Advancement of Science Section: Policy Forum. Available at: <https://science.sciencemag.org/content/320/5878/877>. Cited in page 11.

BERG, H.; PROEBSTING, T. A. Hanson's Automated Market Maker. *Journal of Prediction Markets*, v. 3, n. 1, p. 45–59, 2009. Publisher: University of Buckingham Press. Available at: <https://ideas.repec.org/a/buc/jpredm/v3y2009i1p45-59.html>. Cited in pages 25, 31, and 32.

BICKEL, J. Some Comparisons among Quadratic, Spherical, and Logarithmic Scoring Rules. *Decision Analysis*, v. 4, p. 49–65, jun. 2007. Cited in page 13.

BRAHMA, A. et al. A bayesian market maker. In: *Proceedings of the 13th ACM Conference on Electronic Commerce - EC '12*. Valencia, Spain: ACM Press, 2012. p. 215. ISBN 978-1-4503-1415-2. Available at: <http://dl.acm.org/citation.cfm?doid=2229012.2229031>. Cited in page 12.

CHEN, Y.; PENNOCK, D. M. A Utility Framework for Bounded-Loss Market Makers. *arXiv:1206.5252 [cs, q-fin]*, jun. 2012. ArXiv: 1206.5252. Available at: <http://arxiv.org/abs/1206.5252>. Cited in page 14.

COWGILL, B.; ZITZEWITZ, E. Corporate Prediction Markets: Evidence from Google, Ford, and Firm X *. *The Review of Economic Studies*, v. 82, n. 4, p. 1309–1341, oct. 2015. ISSN 0034-6527. Available at: <https://doi.org/10.1093/restud/rdv014>. Cited in page 11.

DAS, R. et al. Agent-Human Interactions in the Continuous Double Auction. *IJCAI International Joint Conference on Artificial Intelligence*, may 2001. Cited in page 11.

DAS, S. A learning market-maker in the Glosten-Milgrom model. *Quantitative Finance*, v. 5, n. 2, p. 169–180, 2005. Publisher: Taylor & Francis Journals. Available at: <https://ideas.repec.org/a/taf/quantf/v5y2005i2p169-180.html>. Cited in page 20.

EHRENTREICH, N. *Agent-based modeling: the Santa Fe Institute artificial stock market model revisited*. Berlin ; New York: Springer, 2008. (Lecture notes in economics and mathematical systems, 602). OCLC: ocn166386559. ISBN 978-3-540-73878-7. Cited in pages 18, 19, 27, and 41.

FOKIANOS, K.; KEDEM, B. Regression Theory for Categorical Time Series. *Statistical Science*, v. 18, n. 3, p. 357–376, aug. 2003. ISSN 0883-4237, 2168-8745. Publisher: Institute of Mathematical Statistics. Available at: <https://projecteuclid.org/journals/statistical-science/volume-18/issue-3/Regression-Theory-for-Categorical-Time-Series/10.1214/ss/1076102425.full>. Cited in page 22.

HANSON, R. *Idea Futures - The Concept*. 1996. Available at: <http://mason.gmu.edu/~rhanson/ideafutures.html>. Cited in page 11.

HANSON, R. Combinatorial Information Market Design. *Information Systems Frontiers*, v. 5, n. 1, p. 107–119, jan. 2003. ISSN 1572-9419. Available at: <https://doi.org/10.1023/A:1022058209073>. Cited in page 13.

HANSON, R. Logarithmic Market Scoring Rules for Modular Combinatorial Information Aggregation. *The Journal of Prediction Markets*, v. 1, n. 1, p. 3–15, 2007. ISSN 1750-676X. Number: 1. Available at: <http://www.ubplj.org/index.php/jpm/article/view/417>. Cited in pages 11, 14, and 32.

HAYEK, F. A. *The Use of Knowledge in Society*. Rochester, NY, 1945. Available at: <https://papers.ssrn.com/abstract=1505216>. Cited in page 13.

JOHNSON, P. E. *Artificial Stock Market*. 2005. Available at: <http://artstkmkt.sourceforge.net/>. Cited in page 18.

KLINGERT, F. M. A.; MEYER, M. Comparing Prediction Market Mechanisms: An Experiment-Based and Micro Validated Multi-Agent Simulation. *Journal of Artificial Societies and Social Simulation*, v. 21, n. 1, p. 1–7, 2018. Publisher: Journal of Artificial Societies and Social Simulation. Available at: <https://ideas.repec.org/a/jas/jasssj/2016-192-2.html>. Cited in page 12.

LEBARON, B. Building the santa fe artificial stock market. Working Paper, Graduate. In: *School of International Economics and Finance, Brandeis*. [S.l.: s.n.], 2002. p. 1117–1147. Cited in page 18.

OTHMAN, A. et al. A Practical Liquidity-Sensitive Automated Market Maker. *ACM Transactions on Economics and Computation*, v. 1, n. 3, p. 1–25, sep. 2013. ISSN 2167-8375, 2167-8383. Available at: <https://dl.acm.org/doi/10.1145/2509413.2509414>. Cited in pages 11, 15, 16, 17, 20, and 41.

PALMER, R. G. et al. Artificial economic life: a simple model of a stockmarket. *Physica D: Nonlinear Phenomena*, v. 75, n. 1, p. 264–274, aug. 1994. ISSN 0167-2789. Available at: <https://www.sciencedirect.com/science/article/pii/0167278994902879>. Cited in page 18.

PENNOCK, D. M. A dynamic pari-mutuel market for hedging, wagering, and information aggregation. In: *Proceedings of the 5th ACM conference on Electronic commerce*. New York, NY, USA: Association for Computing Machinery, 2004. (EC '04), p. 170–179. ISBN 978-1-58113-771-2. Available at: <https://doi.org/10.1145/988772.988799>. Cited in pages 11 and 13.

PODER360. Homepage, *Poder360 | Notícias do poder e da política*. 2021. Available at: <https://www.poder360.com.br/arquivo/>. Cited in page 22.

ROşU, I. A Dynamic Model of the Limit Order Book. *The Review of Financial Studies*, v. 22, n. 11, p. 4601–4641, nov. 2009. ISSN 0893-9454. Available at: <https://doi.org/10.1093/rfs/hhp011>. Cited in page 11.

SLAMKA, C.; SKIERA, B.; SPANN, M. Prediction Market Performance and Market Liquidity: A Comparison of Automated Market Makers. *IEEE Transactions on Engineering Management*, v. 60, n. 1, p. 169–185, feb. 2013. ISSN 0018-9391, 1558-0040. Available at: <http://ieeexplore.ieee.org/document/6189381/>. Cited in pages 12, 14, 19, and 41.

SMITH, M. A.; PATON, D.; WILLIAMS, L. V. Market Efficiency in Person-to-Person Betting. *Economica*, v. 73, n. 292, p. 673–689, 2006. ISSN 0013-0427. Publisher: [London School of Economics, Wiley, London School of Economics and Political Science, Suntory and Toyota International Centres for Economics and Related Disciplines]. Available at: <https://www.jstor.org/stable/3874062>. Cited in page 16.

WANG, Y. Automated Market Makers for Decentralized Finance (DeFi). *arXiv:2009.01676 [cs, q-fin]*, sep. 2020. ArXiv: 2009.01676. Available at: <http://arxiv.org/abs/2009.01676>. Cited in pages 11 and 13.

WOLFERS, J.; ZITZEWITZ, E. *Interpreting Prediction Market Prices as Probabilities*. [S.l.], 2006. Available at: <https://www.nber.org/papers/w12200>. Cited in page 11.

Appendix

APPENDIX A – Guide to the LMSR-ASM

A.1 What is the LMSR-ASM?

The LMSR-ASM is an adaptation for Prediction Market simulations of Norman Ehrentreich's Java version of the SFI-ASM (NESFI-ASM), as presented in Ehrentreich (2008).

Simulations start with a previously selected number of agents, carrying a predetermined quantity of cash. At each period, agents update their optimal demand for stocks by maximizing a CARA utility (as shown in Ehrentreich (2008)). Agents buy or sell stocks according to the difference between their optimal demand and the number of stocks they currently have. Each agent has a forecast of the probability of the underlying event. This forecast is based on the true probability of the stock plus a random element unique to each agent, following $\mathcal{N}(0, 0.05)$, as defined in Slamka, Skiera and Spann (2013).

A simulation run has the following workflow: either LMSR or LS-LMSR is selected as the Automated Market Maker and, in the first period, initial quantities are created, either to create liquidity, when in LS-LMSR, or to adjust initial prices, in both cases. These initial quantities are *artificial* stocks, they are held by the Market Maker and therefore do not affect profits. The LS-LMSR presents liquidity problems without initial quantities, with the function being undefined for $\boldsymbol{q} = (0, 0)$ and one-unit bet prices being costly at the point of discouraging trades (see Othman et al. (2013)), making these *artificial* stocks necessary to kickstart the market. Regarding price adjustment, initial stocks are created until the point when first-unit bets reach the determined value. Stock creation is a practical way to affect prices without changing the behavior of cost functions.

After this first period, agents can trade stocks according to their own personal forecasts and, at the last period, the underlying event is performed: the Automated Market Maker closes the market paying the agents with winning stocks according to their holdings.

Simulations can also be run in batch mode, using parameter files as described in the Repast guides. A sample batch parameter file is given at the end of this guide.

An in depth explanation of stock and agent behavior is presented in the paper "An Agent-Based Framework for Prediction Markets".

A.2 Dependencies

To run the LMSR-ASM.jar file, you need Java SE 8, the model might not work with later editions. To compile the code, you will also need RepastJ 1.4 and jfreechart 0.9.10.

A.3 Repast *GUI* options and parameters

In the Repast *GUI*, users can set the following variables:

- NumberOfLMSRAgents: the number of agents in the simulation
- MarketMakerMethod: rounding methods for market maker settling. Round to 2 decimal digits, 6 decimal digits or use float values
- AgentType: how agents make purchase decisions and create forecasts, options are Ideal, Logit and Random Walk
- NumberOfPeriods: the duration of simulation
- InterestRate: the risk-free rate of return
- LS-LMSR: changes from LMSR to LS-LMSR
- ShowDisplay: turns simulation graphs on or off
- RecordData: records simulation data to file

The Repast *GUI* also presents the following buttons, which open menus for other settings:

- LMSRAgent: initial cash quantities and agent risk aversion settings
- LMSRStock: a variety of settings regarding event and stock behavior, explained in detail below
- ObserverOptions: choose variables shown in graphs
- RecorderOptions: choose recorded variables and save file name

LMSRStock holds most of the simulation options offered, as follows:

- AlphaLS: sets the α parameter in the LS-LMSR function
- Bliq: sets the b parameter in the LMSR function

- InitialPrice: controls the price of the first "Yes" stock offered
- InitialQuantity: the amount of artificial stocks created at the beginning of the simulation; affects initial liquidity in LS-LMSR
- PeriodShock: sets at which period the probability of the stock changes. When set to 0 the probability never changes during the course of the simulation.
- ProbAfterShock: sets the probability after the shock
- Probability: sets the real probability of the stock
- ProbabilityProcess: a dropdown for choosing between Fixed, Logit and Random Walk probability generating processes

A.4 Class Hierarchy

- AsmModel extends SimModelImpl
- Agent
  - LMSRAgent extends Agent
- Asset
  - LMSRStock extends Asset
- World
- Specialist
- ExecutePeriod
- ObserverOptions
- RecorderOptions
- RecorderParamFileReader

A.5 Class descriptions

A.5.1 World

- Creates agents, resets the simulation and gets wealth levels

A.5.2 Agent

- executeOrder: settles buying and creation or destruction of both types of stock
- getEarningsAndPayTaxes: called in Execute Period, handles agent wealth updating
- setPayout: called in the last period of simulation, pays $1 to holders of the stock conditional to the realization of the underlying event
- setDemandAndSlope: agent behavior is programmed. Agent's forecasts are made and optimal demand is calculated. 'executeOrder' is called at the end of the process
- constrainDemand: prohibits agents from making orders larger than their cash holdings

A.5.3 LMSRAgent

- Used for setting cash and risk aversion properties of agents

A.5.4 LMSRStock

- liquiditySensitiveB: implements LS-LMSR's α
- baseQLMSR: creates artificial stocks
- firstPrice: gets price of buying one stock
- qInitLMSR: sets initial quantity of stocks according to price set in GUI. Creates artificial stocks owned by the Market Maker, until the price of the marginal stock is greater than the price set
- probShock: changes underlying event's probability at chosen period
- updateProbability: handles probability updating between periods according to chosen process

A.5.5 Specialist

- getCostLMSR: calculates the total price of an order using LMSR. The price for buying is rounded up and the price of selling is rounded down.
- getLastPriceLMSR: calculates the price of the last stock in an order, used by agents to determine order quantities
- adjustPricePrediction: handles updating the stock price shown in the display and the price agents use to calculate their demand.

A.5.6 ExecutePeriod

- Handles behavior at each period. Calls for probability updates and agents orders, also updates graphs

A.6 Batch File Example

Here is an example file for running the model in batch mode. Batches can have parameter ranges and these can also be nested, as present in the example. Further explanations can be found in the repast guides.

```
runs: 1

NumberOfLMSRAgents {
   set: 25
}
numberOfPeriods {
   set: 100
}
interestRate {
   set: 0.0
}
memory {
   set: 2500
}
LS_LMSR {
  set_boolean: false
}
showDisplays {
   set_boolean: false
}
recordData {
   set_boolean: true
}
/*
If you want the model to start with identical random seeds, uncomment
the next section.
*/
//RngSeed {
//    set: 1
//}

agentType {
   set: 0
}
alphaLS {
   set: 0.15
}
bLiq {
   set: 87.13
}
initialPrice {
   set: 0.5
}
initialQuantity {
```

```
    set: 50
}
periodShock {
    set: 50.0
}
probAfterShock {
    start: 0.1
    end: 0.9
    incr: 0.1
    {
        runs: 1
        probability {
            start: 0.1
            end: 0.9
            incr: 0.1
        }
    }
}
riskAversion {
    set: 0.3
}
probabilityProcess {
    set: 0
}
recorderOutputFile {
    set_string: Asm1.txt
}
recorderParamFile {
    set_string: recorder.pf
}
}
```