Guide to the LMSR-ASM

Athos Vinicius Cavalcante Carvalho

July 16, 2021

1 What is the LMSR-ASM?

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

Simulations start with a previously selected number of agents, carrying a predetermined quantity of cash. At each period, the agents determine their optimal demand for stocks by maximizing a CARA utility (as shown in Ehrentreich 2008). The 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".

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 Repast J 1.4 and jfreechart 0.9.10.

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

4 Class Hierarchy

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

5 Class descriptions

5.1 World

- Creates agents, resets the simulation and gets wealth levels

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

5.3 LMSRAgent

- Used for setting cash and risk aversion properties of agents

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

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.

5.6 ExecutePeriod

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

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
}
}
```

For any questions you might have, I can be reached at athosvcc@gmail.com.