7CCSMAMF Agent-Based Modelling in Finance
Market Manipulation
Group Grape

Yu-Hsun Wang, Pin-Chi Chen, Hsiao-Ching Liu
Bo Xu, Xuanming Gu

1 Introduction

Financial markets are complex systems that are characterized by the interaction of various agents with different goals, beliefs, and strategies. Understanding how these agents behave influences the market is crucial for policymakers, investors, and traders alike. In recent years, studying the impact of big investors on financial markets has been a big issue. Big investors are institutional investors with a large amount of capital that they can use to influence the market and profit from market trends and fluctuations.

This report presents a model that simulates the behavior of various agents in a financial market, including random traders, chartists, and big investors. The model assumes a closed environment and a limit order book (LOB) mechanism. It also incorporates a momentum strategy that uses Moving Average Convergence Divergence(MACD) as a key technical indicator for implementing trading decisions.

The focus of this report is to examine the manipulation effect of big investors on the market. Specifically, we investigate how big investors use their enormous capital and complex strategies to generate upward momentum and profit from the market. We also examine the impact of big investors on the volatility of the market and their interaction with other agents, such as chartists and random traders.

2 Methodology

2.1 Model assumptions

The model has several assumptions, which include:

1. *Closed environment*: The model assumes that the market is closed, which means all the agents are allocated with a given amount of money and asset (finite) and then start trading,no money is created during the whole process.
2. *Three types of agents*: The model includes three types of agents: random traders, chartists and big investors.
3. *Fixed order quantity*: We assume that every investor sends an equal number of orders (each order's quantity is fixed to 1) for the same asset. The investor must wait until its execution or expiration to send another order.
4. *No regulatory measures or transaction taxes*

2.2 Market mechanism

Our model follows the mechanism of a **limit order book (LOB)**, a popular market mechanism used in financial markets. In a LOB, each trader has a belief in the value

of stocks, randomly set at the beginning. Time is split into ticks, and traders use a pre-determined strategy to put orders on the market in each tick.

Sell orders are matched with buy orders at the stated price or higher, and the two parties exchange stocks and cash. If a sell order isn't fully executed, it goes into the unmatched list. Buy orders go through the same process, but with an opposite bias to match lower prices. The market price and other information are calculated and given to traders in the next tick.

2.3 Market participants and Strategies

Assume there are three types of traders in the market, **random agents**, **chartists** and **big investors**.

2.3.1 Random Agents

Random agents buy, sell and hold an asset with an equal probability. Their actions mainly add liquidity to the market and allow other traders to apply their strategies.

The strategy uses the formulas $P_b(t+1) = P(t) \cdot N(\mu_b, \sigma_t)$ and $P_s(t+1) = P(t) \cdot N(\mu_s, \sigma_t)$ to compute buy and sell order prices, respectively. In these formulas, $P(t)$ represents the current stock price at time t, while $N(\mu_b, \sigma_t)$ and $N(\mu_s, \sigma_t)$ are random numbers drawn from a Gaussian distribution. By setting the mean value of μ_b to 1.01 and μ_s to 0.99, and adjusting the volatility factor with σ_t calculated based on the asset's price volatility in the previous 10 time steps, the trading environment is stimulated, and the volatility of the system is increased. This approach has been observed in the work of Raberto et al. (2001) [1], where a similar volatility factor is used in price formation.

2.3.2 Chartists

Chartists are technical analysts who identify patterns of the stock market to predict stock price movements by identifying patterns. When there are more chartists in the market, the stock tends to have higher volatility.

In this model, *Chartists* utilize the **Moving Average Convergence Divergence (MACD)** as their primary technical analysis tool, recognizing its effectiveness in identifying the underlying trend in asset prices and providing valuable insights into market momentum and direction. The MACD is a powerful indicator, designed to assist traders in making informed decisions by analyzing the relationship between two moving averages and generating signals that can help them stay ahead of market movements. Section 2.4 provides details on how this indicator is used in the strategy.

2.3.3 Big Investors

Big investors have enormous capital and use more complex strategies than other market traders. They aim to profit from market trends and fluctuations by buying at lower prices and selling at higher prices.

The big investor follows a strategy of buying low and selling high, utilizing **MACD** but with a different duration compared to chartists. When the Momentum Strategy signals a buying position, the investor will start to purchase a large proportion of market sell orders (75-95%) until 50% of the strategy duration has elapsed. A brief pause follows, during which there only exist chartists and random agents trading in the market. As it reaches the last 25% and prices stii remain in a high position, the *big investor* gradually sells the orders they hold in a small portion of all the buy orders in the market (20-35%), until all orders have been sold. This strategy allows the investor to purchase at a lower price, generate upward momentum, and sell later for a profit.

2.4 Momentum Strategy

In this model, we are utilizing the **Moving Average Convergence Divergence (MACD)** strategy.

The MACD indicator is calculated by subtracting the 26-day EMA from the 12-day EMA. Therefore, the MACD uses EMAs as its basis for calculation.

2.4.1 EMA

According to Kirkpatrick II and Dahlquist (2010) [2], the exponential moving average (EMA) at time t is defined as:

$$\mathrm{EMA}(t) = (\mathrm{P}(t) * \mathrm{W}) + \mathrm{EMA}(t-1) * (1 - \mathrm{W})$$

where:

- $\mathrm{P}(t)$ is the asset's price at time t
- W is the weighting multiplier, it is computed as: $\mathrm{W} = 2 \div (N_{EMA} + 1)$
- N_{EMA} is the number of days in moving average
- $\mathrm{EMA}(t-1)$ is the exponential moving average at time $t-1$

2.4.2 MACD

The formula for the MACD is:

$$\text{MACD Line} = \text{12-day EMA} - \text{26-day EMA}$$

$$\text{Signal Line} = \text{9-day EMA of the MACD Line}$$

When the MACD line crosses above the signal line, it generates a **buying signal**, indicating a potential upward price trend. On the contrary, it generates a **selling signal**, indicating a potential downward price trend.

2.5 Model parameters

2.5.1 Market Size

The market comprises of 201 investors, which can be classified into three groups: 180 *random agents*, 20 *chartists* and 1 *big investor*.

2.5.2 Initial Wealth of investors

Random agents and *chartists* are defined to hold asset of 80 shares of and money of \$1000. *The big investor* holds a variable amount of wealth, determined by a slider, which can be adjusted to see how it affects the result of the manipulation. *The big-investor's wealth* is represented by a multiple of the initial wealth of the other investors. For example, if the slider is set to 100,000, then *the big investor* would start with 8,000,000 shares and \$100,000,000 of cash, which is 100,000 times the initial wealth of the other investors.

2.5.3 Strategy-related Parameters

The duration of big investor's strategy is parameterized to last for 100 ticks, allowing us to observe its impact on the market and asset pricing.

2.5.4 Order Expiration

When an order reaches its lifespan limit, it will be deleted in the market by the system. The orders in the market expire every 15 ticks, with each tick representing one time step (t+1).

2.5.5 Depletion Behavior

When an investor's stock inventory drops to zero, they have a 25% chance of submitting a buy order to acquire more shares, while the remaining 75% of the time

they hold their current position. Likewise, when an investor runs out of available funds, they have a 25% chance of submitting a sell order to liquidate their current holdings, with a 75% chance of holding their position.

2.5.6 Pseudo Code

Algorithm 1 Market Simulation with Limit Order Book

```
Initialize the simulation environment with the LOB
while simulation not complete do
    Expire old orders in the LOB
    Random traders place new buy/sell orders in the LOB
    The big investor and the chartists place buy/sell orders in the LOB using the MACD indicator
    Match and execute orders in the LOB based on price-time priority
    Update investor wealth and stock price based on executed trades
    Calculate indicators such as traders' wealth, standard deviation, and trading volume
    Update simulation time
Output final results, including stock price and agent wealth distribution
```

3 Results

We conducted three distinct scenarios to analyze the stock market. The first scenario (S1) involved only random agents trading. The second scenario (S2) included chartists alongside random agents. Finally, the third scenario (S3) incorporated the big investor.

Table 1: Summary of Scenarios

Scenario	Agents	Description
S1	Random	Only random agents
S2	Random and Chartists	Chartists added
S3	Random, Chartists, and Big Investor	Big investor added

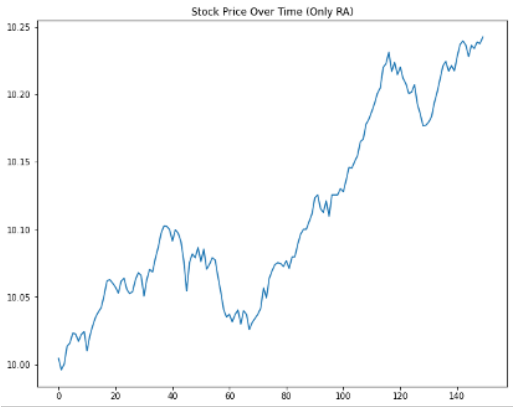
In section 3.1, the diagrams represented random agents trading exclusively (S1).

Moving on to section 3.2, we compared the scenarios of S2 and S3 to observe the impact of the big investor on stock prices and the sigma of the stock price. This allowed us to identify the substantial influence of the big investor on the market.

In section 3.3, we analyzed the sigma distribution across the three different scenarios to determine the volatility range of each scenario.

Finally, in section 3.4, we discussed the return rate of the three types of agents during scenario 3, where the big investor was included. This analysis enabled us to evaluate the performance of each type of agent in the presence of the big investor.

3.1 Scenario 1: Only Random Agents



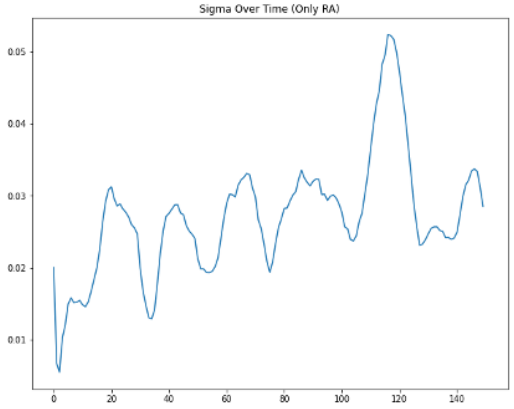


Figure 1, 2. The stock price and sigma generated by random agents in 150 ticks.

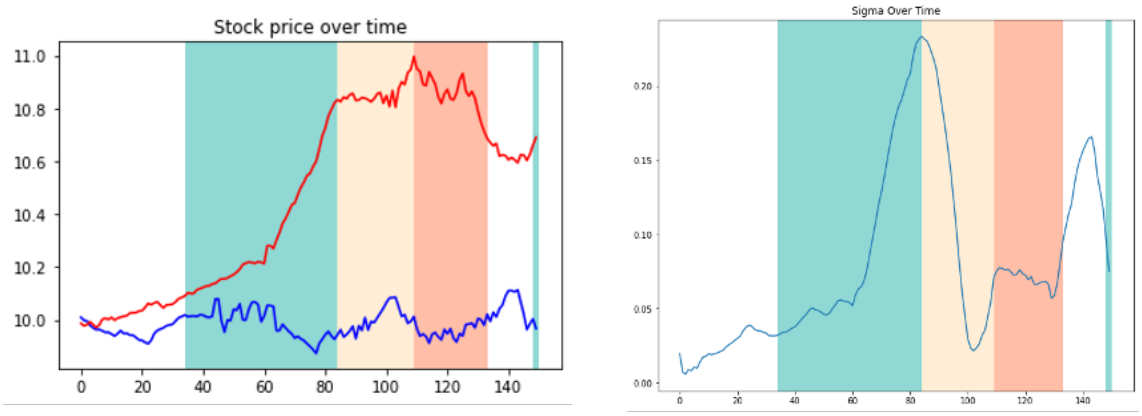
The first scenario aimed to create market liquidity over a 150-tick period by having only random agents send buy or sell orders at random intervals. As a result, the stock price underwent significant fluctuations as the random agents' actions had a direct impact on the market.

To better understand the characteristics of the scenarios involving chartists and the big investor, we would conduct a comparison analysis with the first scenario. This comparison analysis allowed us to identify the key features of scenarios S2 and S3, and to evaluate the impact of adding chartists and the big investor to the market. By comparing the outcomes of these scenarios, we gained insight into the behavior of different types of investors and their effects on the stock price and volatility levels.

3.2 Single Pump-and-Dump Analysis

Figure 3. Compared the stock price over 150 ticks across two different scenarios: S2 (blue line) and S3 (red line).

It demonstrated how the stock price changed significantly after adding a big investor. The big investor influenced this market's stock price, leading to market manipulation. The red line also climbed considerably in the green zone due to the big investor purchases at that time. This was followed by a modest fluctuation in the yellow part, which varied from 85 to approximately 110, and then experienced



a moderate downward trend in the orange section as the result of the big investor's stock sale.

Figure 4. The market's overall sigma over S3

Sigma enabled us to identify the pump-and-dump behavior of the big investor simultaneously. During the pump (green zone), the volatility would upsurge. It would plummet to almost zero when the big investor's strategy reached the static stage and then went up to a certain level when the dump occurred.

3.3 Sigma Analysis

The big investor significantly increased market volatility (sigma) compared to the other two scenarios, which affected the magnitude of stock price movements. We can easily compare the three scenarios with the histograms on the right side.

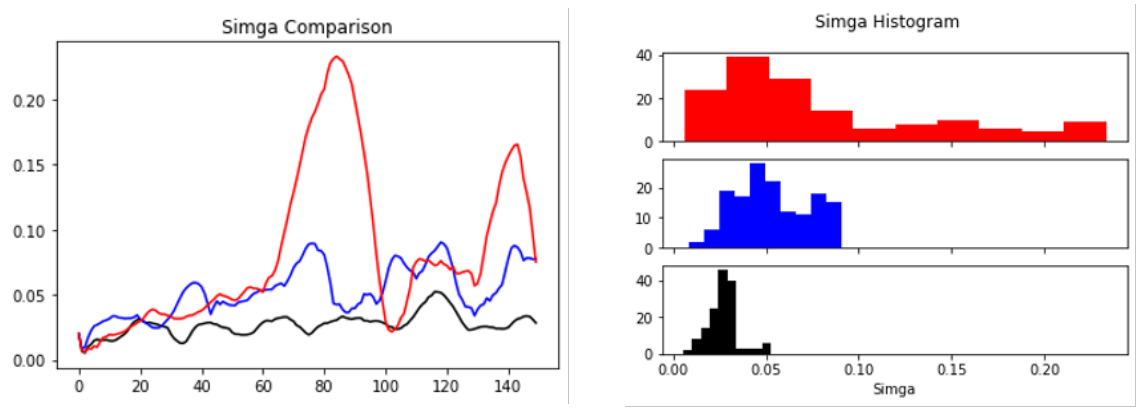


Figure 5, 6. Sigma comparison via line graph and three histograms The red line contained three categories of investors: random agents, chartists, and a big investor in the 150 ticks doing a single pump and dump. The

blue line included random agents and chartists. The black line represented only random agents.

Based on the histogram depicted on the right, it is evident that the sigma of S1, which involves only random agents, is concentrated between 0 and 0.05. However, when chartists are incorporated, the volatility range widens and falls within 0 and 0.09. Subsequently, with the inclusion of the big investor, the sigma distribution takes a right-tailed shape, where the proportion of extreme values increases significantly, and the overall range is between 0 and 0.24. Hence, it is apparent that the presence of the big investor has a substantial impact on the sigma. Thus, it can be inferred that the addition of the big investor plays a crucial role in determining the volatility of the market.

3.4 Return Analysis

In order to create a more robust and comprehensive model, we increased the number of ticks from 150 to 2000, thereby including several pumps and dumps in our analysis. This enabled us to observe the market over a more extended period and obtain a more accurate representation of the behavior of different investors.

Table 2: Return Rate of Different Types of Investors in S3 over 2000 ticks

Investor Type	**Return Rate**
Random Agents	-29.05%
Chartists	109.03%
The Big Investor	157.06%

Upon analysis of the resulting data, we observed that the big investor outperformed the other investors by a significant margin. The big investor's return rate of 157.06% was notably higher than that of the chartists, who had a return rate of 109.03%. Additionally, the return rate of the random agents was negative at -29.05%, indicating that they performed poorly in this scenario.

Our findings suggest that the big investor had a considerable advantage over the other investors, possibly due to their greater financial resources and market influence. The big investor's significant return rate may have been a result of their ability to influence the market through their trading activity, leading to an increased demand for the stock and driving up prices. This, in turn, may have allowed the big investor to sell their shares at a higher price, resulting in a substantial profit.

3.5 Sensitivity Analysis

In this section, we changed three parameters(buy_ratio, sell_ratio and b_multiplier) and found that changing of these parameters had no effect on the normal behavior

of model. We can conclude that the model is robust to these parameters.

4 Conclusion

4.1 Potential Mitigation Strategies

To conclude, market manipulation can cause significant damage to the financial market and individual investors. In order to prevent market manipulation, we come up with multiple strategies in two aspects. First, enhanced market surveillance is fundamental to avoid the occurrence of manipulation. To achieve that, we mainly implement advanced data analytics and artificial intelligence techniques to monitor market activities in real time. Additionally, detecting unusual trading patterns is also an efficient way to help us identify potential pump-and-dump schemes early.

Secondly, market regulators should figure out ways to Increase transparency in financial markets, especially in emerging areas. Encouraging greater transparency in financial markets can be realized by mandating timely and accurate disclosure of relevant information and promoting the use of secure and transparent platforms for conducting market transactions.

4.2 Future work

Despite we have built the model and identified some key characteristics by running the model, there are many potential ideas which need to be discarded.

Future research can be done to explore the dynamics of market manipulation by looking into how variations of strategies of big investors can have an effect on the market, and then evaluate the efficacy of these strategies. Moreover, new tactics and techniques that can be used by market manipulators to stay ahead of regulatory efforts need to be identified. The last important issue for future work is how manipulation responds to regulation.

5 Bibliography

References

[1] Raberto, M., Cincotti, S., Focardi, S. M., & Marchesi, M. (2001). Agent-based simulation of a financial market. Physica A: Statistical Mechanics and Its Applications, 299(1–2), 319–327. https://doi.org/10.1016/S0378-4371(01)00312-0

[2] Kirkpatrick II, C. D. and Dahlquist, J. A. (2010) "Technical analysis: the complete resource for financial market technicians". FT press.

Appendix A

5.1 Python Code

https://github.com/AuroraLiu3230/Market-Manipulation-Model

```
import random
import statistics
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
class LimitOrderBook:
    def __init__(self):
        self.bids = []
        self.asks = []
        self.stock_price = 10
        self.count = 0
        self.stepid = 0
        self.Close = [10]
    def insert_order(self, order):
        if order.type == 'buy':
            self.bids.append(order)
            self.bids.sort(key=lambda x: x.price, reverse=
                True)
        elif order.type == 'sell':
            self.asks.append(order)
            self.asks.sort(key=lambda x: x.price)
    def match_orders(self):
        self.count = 0
        transaction_price = self.stock_price
        tranList = []
        while self.bids and self.asks:
            for i in self.bids:
                if self.stepid - i.stepid == 15:
                    if i.who <= 180:
                        for x in range(1,181):
                            if lob_agent.trader_info[x]== i.
                                who:
                                lob_agent.trader_info[x]['
                                    money'] += i.price

                    if i.who > 180 and i.who <= 200:
                        for y in range(181,201):
                            if lob_agent.c.trader_info[y]==
                                i.who:
                                lob_agent.c.trader_info[y]['
```

```
                                    money'] += i.price
37
38                      if i.who == 201:
39                          for z in range(201,202):
40                              if lob_agent.b.trader_info[z] ==
                                    i.who:
41                                  lob_agent.b.trader_info[z]['
                                        money'] += i.price
42                      self.bids.remove(i)
43              for j in self.asks:
44                  if self.stepid - j.stepid == 15:
45                      self.asks.remove(j)
46              if self.bids[0].price >= self.asks[0].price:
47                  transaction_price = (self.bids[0].price +
                        self.asks[0].price) / 2
48
49
50                  if self.bids[0].who <= 180:
51                      lob_agent.trader_info[self.bids[0].who][
                            'shares'] += 1
52                      lob_agent.trader_info[self.bids[0].who][
                            'money'] += (self.bids[0].price -
                            transaction_price)
53                  if self.asks[0].who <= 180:
54                      lob_agent.trader_info[self.asks[0].who][
                            'money'] += transaction_price
55                  if self.bids[0].who > 180 and self.bids[0].
                        who <= 200:
56                      lob_agent.c.trader_info[self.bids[0].who
                            ]['shares'] += 1
57                      lob_agent.c.trader_info[self.bids[0].who
                            ]['money'] += (self.bids[0].price -
                            transaction_price)
58                  if self.asks[0].who > 180 and self.asks[0].
                        who <= 200:
59                      lob_agent.c.trader_info[self.asks[0].who
                            ]['money'] += transaction_price
60                  if self.bids[0].who == 201:
61                      lob_agent.b.trader_info[self.bids[0].who
                            ]['shares'] += 1
62                      lob_agent.b.trader_info[self.bids[0].who
                            ]['money'] += (self.bids[0].price -
                            transaction_price)
63                  if self.asks[0].who == 201:
64                      lob_agent.b.trader_info[self.asks[0].who
                            ]['money'] += transaction_price
```

```
65                  self.stock_price = transaction_price
66                  self.count += 1
67                  transaction_quantity = min(self.bids[0].
                        quantity, self.asks[0].quantity)
68                  self.bids[0].quantity -=
                        transaction_quantity
69                  self.asks[0].quantity -=
                        transaction_quantity
70                  if self.bids[0].quantity == 0:
71                      self.bids.pop(0)
72                  if self.asks[0].quantity == 0:
73                      self.asks.pop(0)
74                  tranList.append(transaction_price)
75              else:
76                  if tranList != []:
77                      close = statistics.median(tranList)
78                      self.Close.append(close)
79                  break
80          self.stepid += 1
81      def print_order_book(self):
82          print("Bids:")
83          for order in self.bids:
84              print(order)
85          print("Asks:")
86          for order in self.asks:
87              print(order)
88          print("Stock Price:", self.stock_price)
89  class Order:
90      def __init__(self, order_type, quantity, price, who,
            stepid):
91          self.type = order_type
92          self.quantity = quantity
93          self.price = price
94          self.who = who
95          self.stepid = stepid
96      def __str__(self):
97          return f"{self.type} {self.quantity}@{self.price} {
                self.who} {self.stepid}"
98  class Chartists:
99      def __init__(self, num_traders=20):
100         self.lob = LimitOrderBook()
101         self.num_traders = num_traders
102         self.close = 0
103         self.asks = 0
104         self.bids = 0
105         self.macd_signal = []
```

```
106         self.trader_info = {}
107         for i in range(181, (181+num_traders)):
108             self.trader_info[i] = {
109                 'shares': 80,
110                 'money': 1000
111             }
112     def run(self, fast_period=12, slow_period=26,
            signal_period=9):
113         close_price = np.array(lob_agent.StockPrices)
114         ema_fast = pd.Series(close_price).ewm(span=
                fast_period, min_periods=fast_period).mean()
115         ema_slow = pd.Series(close_price).ewm(span=
                slow_period, min_periods=slow_period).mean()
116         macd = ema_fast - ema_slow
117         signal = macd.ewm(span=signal_period, min_periods=
                signal_period).mean()
118         self.macd_signal = np.where(macd > signal, 1, 0)
119 class BigInvestor:
120     def __init__(self, buy_ratio=0.75, sell_ratio=0.2,
            mutiplier = 100000):
121         self.buy_ratio = buy_ratio
122         self.sell_ratio = sell_ratio
123         self.strategy_duration = 100
124         self.strategy_tick = 1
125         self.c = Chartists()
126         self.lob = LimitOrderBook()
127         self.close = 0
128         self.asks = 0
129         self.bids = 0
130         self.la = 0
131         self.lb = 0
132         self.macd_signal = []
133         self.trader_info = {}
134         self.asset = []
135         for j in range(201, 202):
136             self.trader_info[j] = {
137                 'shares': 80 * mutiplier,
138                 'money': 1000 * mutiplier
139         }
140     def run(self, fast_period=12, slow_period=26,
            signal_period=9):
141         close_price = np.array(lob_agent.StockPrices)
142         ema_fast = pd.Series(close_price).ewm(span=
                fast_period, min_periods=fast_period).mean()
143         ema_slow = pd.Series(close_price).ewm(span=
                slow_period, min_periods=slow_period).mean()
```

```
144             macd = ema_fast - ema_slow
145             signal = macd.ewm(span=signal_period, min_periods=
                    signal_period).mean()
146             self.macd_signal = np.where(macd > signal, 1, 0)
147             self.asset.append(self.trader_info[201]['shares']*
                    close_price[-1]+self.trader_info[201]['money'])
148
149
150
151
152 class RandomTrader:
153     def __init__(self, initial_price, time_steps,
            num_RandomTraders, big_strategy = True, c_strategy=
            True, sellfirst_strategy=True, b_buy_ratio=0.75,
            b_sell_ratio=0.2, b_multiplier = 100000):
154             self.b_buy_ratio = b_buy_ratio
155             self.b_sell_ratio = b_sell_ratio
156             self.sellfirst_strategy = sellfirst_strategy
157             self.c_strategy = c_strategy
158             self.big_strategy = big_strategy
159             self.P0 = initial_price
160             self.T = time_steps
161             self.lob = LimitOrderBook()
162             self.c = Chartists()
163             self.b = BigInvestor(buy_ratio=b_buy_ratio,
                    sell_ratio=b_sell_ratio, mutiplier = b_multiplier
                    )
164             self.lob.transactions = []
165             self.sigma = 0.02 / np.sqrt(252)
166             self.StockPrices = []
167             self.num_traders = num_RandomTraders
168             self.stepid = 0
169             self.flag1 = 2
170             self.flag2 = 2
171             self.buyfirst = 1
172             self.sellfirst = 1
173             self.trader_info = {}
174             self.lala = []
175             self.BGtimedivided = [0]
176             self.Ltimedivided = [0]
177             self.bgcolor = ['#FFFFFF']
178             for i in range(1, num_RandomTraders+1):
179                 self.trader_info[i] = {
180                     'shares': 80,
181                     'money': 1000
182                 }
```

```
183
184         self.sigmaList_stat = []
185         self.r_wealth = []
186         self.b_wealth = []
187         self.c_wealth = []
188         self.LogReturn = [0]
189         self.Volume = []
190
191     def run(self):
192
193         for t in range(self.T):
194             self.stepid = t
195             if t > 0:
196                 prices = [transaction for transaction in
                        self.lob.Close[-20:]]
197                 if len(prices) < 2:
198                     self.sigma = 0.02 / np.sqrt(252)
199                 else:
200                     self.sigma = np.std(prices) / np.mean(
                            prices)
201             mu_b = 1.01
202             mu_s = 0.99
203             actions = ['buy', 'sell', 'hold']
204             trends = []
205             for i in range(1, self.num_traders+1):
206                 if self.trader_info[i]['shares'] == 0:
207                     trends.append([1/4, 0, 3/4])
208                 elif self.trader_info[i]['money'] == 0:
209                     trends.append([0, 1/4, 3/4])
210                 else:
211                     trends.append([1/3, 1/3, 1/3])
212             for i in range(1, self.num_traders+1):
213                 action = random.choices(actions, weights=
                        trends[i-1])[0]
214                 if action == 'buy':
215                     price_b = self.P0 * random.gauss(mu_b,
                            self.sigma)
216                     if self.trader_info[i]['money'] >=
                            price_b:
217                         self.lob.insert_order(Order('buy',
                                1, price_b, i, self.stepid))
218                         self.trader_info[i]['money'] -=
                                price_b
219
220                 elif action == 'sell':
221                     if self.trader_info[i]['shares'] > 0:
```

```
222                         price_s = self.P0 * random.gauss(
                                mu_s, self.sigma)
223                         self.lob.insert_order(Order('sell',
                                1, price_s, i, self.stepid))
224                         self.trader_info[i]['shares'] -= 1
225             if t >= 26:
226                 if self.c_strategy == True:
227                     self.c.run()
228                     for i in range(self.num_traders+1, (self
                            .num_traders+1 + self.c.num_traders))
                            :
229                         price_b = self.c.asks
230                         if  self.c.macd_signal[-1] == 1 and
                                self.c.macd_signal[-2] == 0 and
                                self.c.trader_info[i]['money'] >=
                                 price_b:
231                             self.lob.insert_order(Order('buy
                                    ', 1, price_b, i, self.stepid
                                    ))
232                             self.c.trader_info[i]['money']
                                    -= price_b
233                         elif self.c.macd_signal[-1] == 0 and
                                 self.c.macd_signal[-2] == 1 and
                                self.c.trader_info[i]['shares'] >
                                 0:
234                             price_s = self.c.bids
235                             self.lob.insert_order(Order('
                                    sell', 1, price_s, i, self.
                                    stepid))
236                             self.c.trader_info[i]['shares']
                                    -= 1
237                 if self.big_strategy == True:
238                     self.b.run()
239                     if (self.b.macd_signal[-1] == 1 and self
                            .b.macd_signal[-2] == 0 and self.
                            flag2 == 2) or self.flag1 == 1:
240                         self.flag1 = 1
241                         self.lala.append(t)
242                         initial_price = self.lob.Close[-1]
243                         price_b = self.b.asks
244                         if self.buyfirst <= self.b.
                                strategy_duration * 0.5:
245                             self.BGtimedivided.append(t)
246                             self.bgcolor.append('#20B2AA')
247                             quantity_b = self.b.buy_ratio *
                                    self.b.la
```

```
248                     for i in range(int(quantity_b)):
249                         if self.b.trader_info[self.
                               num_traders+self.c.
                               num_traders+1]['money']
                              >= price_b:
250                             self.lob.insert_order(
                                   Order('buy', 1,
                                   price_b, self.
                                   num_traders+self.c.
                                   num_traders+1, self.
                                   stepid))
251                     self.b.strategy_tick += 1
252                     self.buyfirst += 1
253                 elif self.b.strategy_duration * 0.5
                      < self.buyfirst and self.buyfirst
                       <= self.b.strategy_duration *
                      0.75 :
254                     self.BGtimedivided.append(t)
255                     self.bgcolor.append('#FFDEAD')
256                     self.b.strategy_tick += 1
257                     self.buyfirst += 1
258                 else:
259                     if self.buyfirst < self.b.
                          strategy_duration:
260                         self.BGtimedivided.append(t)
261                         self.bgcolor.append('#FF7F50
                              ')
262                         self.b.strategy_tick += 1
263                         self.buyfirst += 1
264                         if self.b.lb != 0 and self.b
                              .close[-1] >=
                              initial_price:
265                             quantity_s = self.b.
                                  sell_ratio * self.b.
                                  lb
266
267                             for i in range(int(
                                  quantity_s)):
268                                 if self.b.
                                      trader_info[self.
                                      num_traders+self.
                                      c.num_traders+1][
                                      'shares'] > 0:
269                                     price_s = self.b
                                          .bids
270                                     self.lob.
```

```
                                        insert_order(
                                        Order('sell',
                                         1, price_s,
                                        self.
                                        num_traders+
                                        self.c.
                                        num_traders
                                        +1, self.
                                        stepid))
271                                     self.b.
                                        trader_info[
                                        self.
                                        num_traders+
                                        self.c.
                                        num_traders
                                        +1]['shares']
                                         -= 1
272                                 else:
273                                     self.
                                        BGtimedivided
                                        .append(t)
274                                     self.bgcolor.
                                        append('#
                                        FF7F50')
275                                     pass
276                 if self.buyfirst == 100:
277                     self.flag1 = 2
278                     self.buyfirst = 0
279                     self.lala.append("a")
280                     self.lala.append(t)
281                     self.BGtimedivided.append(t)
282                     self.bgcolor.append('#FF7F50')
283             elif ((self.b.macd_signal[-1] == 0 and
                    self.b.macd_signal[-2] == 1 and self.
                    flag1 == 2) or self.flag2 == 1) and
                    self.sellfirst_strategy:
284                 self.flag2 = 1
285                 initial_price = self.lob.Close[-1]
286                 if self.sellfirst <= self.b.
                        strategy_duration * 0.5:
287                     quantity_s = self.b.buy_ratio *
                            self.b.lb
288                     for i in range(int(quantity_s)):
289                         if self.b.trader_info[self.
                                num_traders+self.c.
                                num_traders+1]['shares']
```

```
                    > 0:
290                     price_s = self.b.bids
291                     self.lob.insert_order(
                            Order('sell', 1,
                            price_s, self.
                            num_traders+self.c.
                            num_traders+1, self.
                            stepid))
292                     self.b.trader_info[self.
                            num_traders+self.c.
                            num_traders+1]['
                            shares'] -= 1
293                 self.b.strategy_tick += 1
294                 self.sellfirst += 1
295             elif self.b.strategy_duration * 0.5
                    < self.sellfirst and self.
                    sellfirst <= self.b.
                    strategy_duration * 0.75 :
296                 self.b.strategy_tick += 1
297                 self.sellfirst += 1
298             else:
299                 if self.b.la != 0 and self.b.
                        close[-1] <= initial_price:
300                     quantity_b = self.b.
                            sell_ratio * self.b.la
301                     price_b = self.b.asks
302                     for i in range(int(
                            quantity_b)):
303                         if self.b.trader_info[
                                self.num_traders+self
                                .c.num_traders+1]['
                                money'] >= price_b:
304                             self.lob.
                                    insert_order(
                                    Order('buy', 1,
                                    price_b, self.
                                    num_traders+self.
                                    c.num_traders+1,
                                    self.stepid))
305                         else:
306                             pass
307                     self.b.strategy_tick += 1
308                     self.sellfirst += 1
309             if self.sellfirst == 100:
310                 self.flag2 = 2
311                 self.sellfirst = 0
```

```
312                             self.lala.append("b")
313                             self.lala.append(t)
314                             self.BGtimedivided.append(t)
315                             self.bgcolor.append('#ebebeb')
316                             if t+100 < self.T:
317                                 self.BGtimedivided.append(t
                                        +100)
318                                 self.bgcolor.append('#FFFFFF
                                        ')
319                         else:
320                             self.BGtimedivided.append(t)
321                             self.bgcolor.append('#FFFFFF')
322             self.lob.match_orders()
323             self.P0 = self.lob.stock_price
324             self.c.close = self.lob.Close
325             if self.lob.asks == []:
326                 self.c.asks = self.lob.Close[-1]
327                 self.b.asks = self.lob.Close[-1]
328             else:
329                 self.c.asks = self.lob.asks[0].price
330                 self.b.asks = self.lob.asks[int(len(self.lob
                        .asks)*0.4)].price
331             if self.lob.bids == []:
332                 self.c.bids = self.lob.Close[-1]
333                 self.b.bids = self.lob.Close[-1]
334             else:
335                 self.c.bids = self.lob.bids[0].price
336                 self.b.bids = self.lob.bids[int(len(self.lob
                        .bids)*0.15)].price
337             self.b.close = self.lob.Close
338             self.b.la = len(self.lob.asks)
339             self.b.lb = len(self.lob.bids)
340             self.StockPrices.append(self.P0)
341             prices = [transaction for transaction in self.
                    StockPrices[-20:]]
342             if len(prices) < 2:
343                 self.sigma = 0.02 / np.sqrt(252)
344             else:
345                 self.sigma = np.std(prices) / np.mean(prices
                        )
346
347             self.sigmaList_stat.append(self.sigma)
348             if len(self.StockPrices) >= 2:
349                 self.LogReturn.append(np.log(self.P0)-np.log
                        (self.StockPrices[-2]))
350             ttl_r_shares = 0
```

```
351                 ttl_r_money = 0
352                 for i in range(1, self.num_traders+1):
353                     ttl_r_shares +=  self.trader_info[i]['shares
                        ']
354                     ttl_r_money += self.trader_info[i]['money']
355                 self.r_wealth.append(ttl_r_shares*self.P0 +
                    ttl_r_money)
356                 ttl_c_shares = 0
357                 ttl_c_money = 0
358                 for i in range(self.num_traders+1, self.
                    num_traders+1+self.c.num_traders):
359                     ttl_c_shares +=  self.c.trader_info[i]['
                        shares']
360                     ttl_c_money += self.c.trader_info[i]['money'
                        ]
361                 self.c_wealth.append(ttl_c_shares*self.P0 +
                    ttl_c_money)
362                 self.b_wealth.append(self.b.trader_info[self.
                    num_traders+self.c.num_traders+1]['shares']*
                    self.P0+ self.b.trader_info[self.num_traders+
                    self.c.num_traders+1]['money'])
363
364                 self.Volume.append(self.lob.count)
365             self.df = pd.DataFrame({
366                 "Close": self.StockPrices,
367                 "Volume": self.Volume,
368                 "Sigma": np.array(self.sigmaList_stat) * np.sqrt
                    (252),
369                 "Variance": (np.array(self.sigmaList_stat) * np.
                    sqrt(252))**2,
370                 "LogReturn": np.array(self.LogReturn),
371                 "Random␣Agents": self.r_wealth,
372                 "Chartists": self.c_wealth,
373                 "Big␣Investor": self.b_wealth
374             })
375             self.BGtimedivided.append(self.T)
376             self.bgcolor.append('#FFFFFF')
377 lob_agent = RandomTrader(initial_price=10, time_steps=2000,
        num_RandomTraders=180,
378                              big_strategy=False, c_strategy=
                                 False, sellfirst_strategy=False,
379                              b_buy_ratio=0.75, b_sell_ratio=0.2,
                                  b_multiplier = 100000)
380 lob_agent.run()
381 plt.figure(figsize=(10,8))
382 plt.title("Stock␣Price␣Over␣Time␣(Only␣RA)")
```

```
383 plt.plot(lob_agent.StockPrices)
384 time = lob_agent.BGtimedivided
385 color = lob_agent.bgcolor
386 for idx in range(len(time)-1):
387     plt.axvspan(time[idx], time[idx+1], facecolor=color[idx
        ], alpha=0.5)
388 plt.show()
389
390
391 plt.figure(figsize=(10,8))
392 plt.title("Sigma Over Time (Only RA)")
393 plt.plot(lob_agent.df["Sigma"])
394 for idx in range(len(time)-1):
395     plt.axvspan(time[idx], time[idx+1], facecolor=color[idx
        ], alpha=0.5)
396 plt.show()
397 df1 = lob_agent.df
398 sigma_1 = lob_agent.df['Sigma']
399 lob_agent = RandomTrader(initial_price=10, time_steps=2000,
    num_RandomTraders=180,
400                          big_strategy=False, c_strategy=True
                                , sellfirst_strategy=False,
401                          b_buy_ratio=0.75, b_sell_ratio=0.2,
                                b_multiplier = 100000)
402 lob_agent.run()
403 plt.figure(figsize=(10,8))
404 plt.title("Stock Price Over Time (RA + Chartists)")
405 plt.plot(lob_agent.StockPrices)
406 time = lob_agent.BGtimedivided
407 color = lob_agent.bgcolor
408 for idx in range(len(time)-1):
409     plt.axvspan(time[idx], time[idx+1], facecolor=color[idx
        ], alpha=0.5)
410 plt.show()
411 plt.figure(figsize=(10,8))
412 plt.title("Sigma Over Time  (RA + Chartists)")
413 plt.plot(lob_agent.df["Sigma"])
414 for idx in range(len(time)-1):
415     plt.axvspan(time[idx], time[idx+1], facecolor=color[idx
        ], alpha=0.5)
416 plt.show()
417 df2= lob_agent.df
418 sigma_2 = lob_agent.df['Sigma']
419 lob_agent = RandomTrader(initial_price=10, time_steps=2000,
    num_RandomTraders=180,
420                          big_strategy=True, c_strategy=True,
```

```
                                    sellfirst_strategy=False,
421                                 b_buy_ratio=0.75, b_sell_ratio=0.2,
                                    b_multiplier = 100000)
422 lob_agent.run()
423 plt.figure(figsize=(15,8))
424 plt.title("Stock Price Over Time (With the Big Investor)")
425 plt.plot(lob_agent.StockPrices)
426 time = lob_agent.BGtimedivided
427 color = lob_agent.bgcolor
428 for idx in range(len(time)-1):
429     plt.axvspan(time[idx], time[idx+1], facecolor=color[idx
        ], alpha=0.5)
430 plt.show()
431 plt.figure(figsize=(10,8))
432 plt.title("Sigma Over Time")
433 plt.plot(lob_agent.df["Sigma"])
434 for idx in range(len(time)-1):
435     plt.axvspan(time[idx], time[idx+1], facecolor=color[idx
        ], alpha=0.5)
436 plt.show()
437 df3 = lob_agent.df
438 sigma_3 = lob_agent.df['Sigma']
439 plt.figure(figsize=(10,8))
440 plt.title("Asset of the Big Investor Over Time (With the Big
    Investor)")
441 plt.plot(lob_agent.df["Big Investor"])
442 for idx in range(len(time)-1):
443     plt.axvspan(time[idx], time[idx+1], facecolor=color[idx
        ], alpha=0.5)
444 plt.show()
445 print("Simulation data")
446 df=lob_agent.df
447 print(df)
448 plt.plot(sigma_1, c='black')
449 plt.plot(sigma_2, c='blue')
450 plt.plot(sigma_3, c='red')
451 plt.title("Simga Comparison")
452 plt.show()
453 fig, (ax3, ax2, ax1) = plt.subplots(3, sharex=True)
454 fig.suptitle('Simga Histogram')
455 ax3.hist(sigma_3, color='red')
456 ax2.hist(sigma_2, color='blue')
457 ax1.hist(sigma_1, color='black')
458 plt.xlabel("Simga")
459 plt.ylabel=("Freq")
460 plt.show()
```

```
461
462 return_df = pd.DataFrame({
463     "Random Agents":[(df["Random Agents"][len(df["Random
        Agents"])-1]-df["Random Agents"][0])/df["Random
        Agents"][0]],
464     "Chartists":[(df["Chartists"][len(df["Chartists"])-1]-df
        ["Chartists"][0])/df["Chartists"][0]],
465     "Big Investor":[(df["Big Investor"][len(df["Big Investor
        "])-1]-df["Big Investor"][0])/df["Big Investor"][0]],
466 },index=["Return rate"])
467 print(return_df)
```

Appendix B

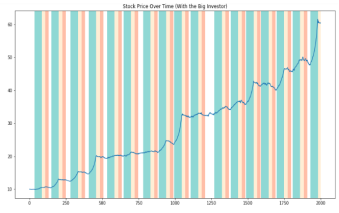
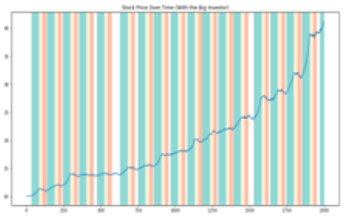
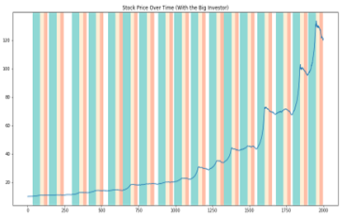




Figure 7, 8 , 9. Different output of stock price over time when buy ratio is 0.75 , 0.8 , 0.85 in the premise that sell ratio remains the same.

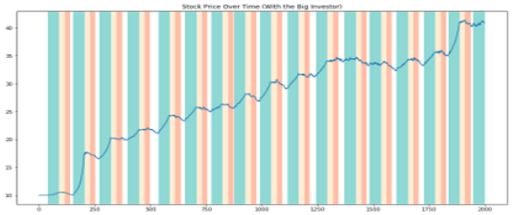
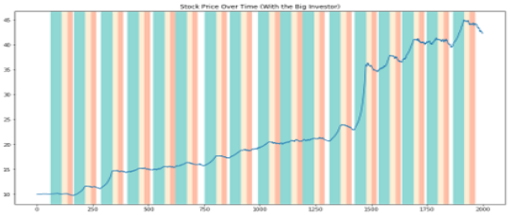



Figure 10, 11. Different output of stock price over time when sell ratio is 0.25 , 0.3 in the premise that buy ratio remains the same.

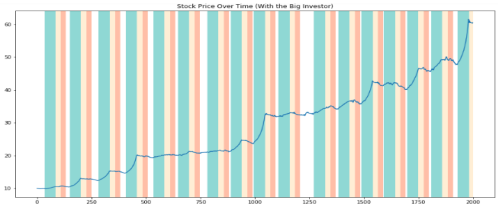
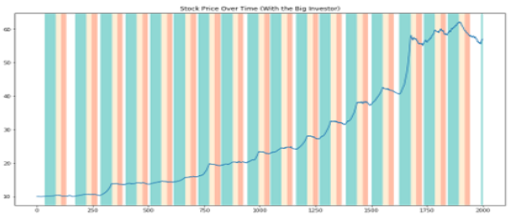



Figure 12, 13. Different output of stock price over time when big investor multiplier is 100000 , 1000000 in the premise that buy ratio and sell ratio remain the same.