University of Zurich[UZH]

Department of Business Administration
Chair for Quantitative Business Administration

Master Thesis

An Agent-Based Model of High-Frequency Trading

Supervisors:

Prof. Dr. Karl Schmedders
Dr. Gregor Reich

Author:

Nicole B. Miserez

Field of Study:	Banking & Finance
Student ID:	12-735-445
Address:	Herman-Greulich-Strasse 48, 8004 Zürich
E-Mail:	nicolebarbara.lipsky@uzh.ch
Submission Date:	August 7, 2017

Executive Summary

The aim of this Master thesis is to build an extensible, robust implementation of the agent-based model presented in "Rock around the Clock: An Agent-Based Model of Low- and High-Frequency Trading" by Jacob Leal et al. (2016). This model is used to study the interplay of low- and high-frequency traders and to observe their effect on asset price dynamics.

After presenting an overview of the current existing relevant literature, I thoroughly discuss the setup of the model. The main objective of this thesis is set on a reusable implementation of the original model, which reproduces the original results, but is also open to various extensions and modifications in different areas. This enables me to study the model's strengths and weaknesses.

Furthermore, the thesis contains a detailed description of the implementation design with graphical illustrations of the architecture. Where necessary, I elaborate design decisions.

The closing section presents the results, and draws conclusion on the validity of the original model as well as my implementation thereof. I propose some possible extensions to test the model's elasticity towards marginal changes in underlying assumptions.

Contents

List of Figures

List of Tables

Listings

Acronyms

ABM	**A**gent-**B**ased **M**odel
DOM	**D**ocument **O**bject **M**odel
HF	**H**igh **F**requency
LF	**L**ow **F**requency
LOB	**L**imit **O**rder **B**ook
UML	**U**nified **M**odeling **L**anguage

1 Introduction

For the last few years, computer algorithms have increasingly replaced humans in the stock exchanges, trading equity in a millisecond environment. Despite the fact that high-frequency trading emerged within quite a short time span, it is already prevalent in electronic stock markets around the globe. Although concrete numbers do not exist, estimations suggest that around 60 to 70 percent of the dollar trading volume can be imputed to high-frequency (HF) trading activity (Carrion 2013, Westerhoff 2008). However, research in this field is still in its early stages. This is due to, firstly, its sudden emergence, and secondly, the fact that high-quality data with sufficient information on trades in the millisecond environment were not existent until very recently. Therefore, in order to understand its power and its opportunities, but also its risks and implications, many researchers aspired to advance the relevant research.

One possibility to investigate in this area is to make use of an agent-based model (ABM). ABMs can be used e.g. to study the mechanics of HF trading and its effects on market quality measures, as did Jacob Leal et al. (2016). In this Master thesis, I replicate this ABM of high- and low-frequency traders. In this model, low-frequency (LF) traders choose between two strategies and can sell or buy in each period. HF traders are bounded in a zero-intelligence framework in which they can only follow one strategy. I thoroughly discuss the modeling framework and parameters by Jacob Leal et al. (2016) and implement the model in a way that it is general enough to be extensible. Furthermore, I incorporate both inputs that I gain from other papers and alternative methodologies that are my own ideas. Ultimately, I validate these extensions.

My findings include that some of the extensions do fulfill the validation criteria and are thus suitable as a financial market model. Others, however, despite only differing marginally to the original model, not only fall short of fulfilling the valida-

tion criteria, but even cause the simulation to completely fall apart. This behavior demonstrates a major shortcoming of this agent-based model: its fragility towards minor changes in its underlying assumptions.

The paper proceeds as follows. Section 2 reviews relevant literature, which contains either empirical studies on HF trading or ABMs thereof. Section 3 describes the original ABM and scrutinizes its design. Section 4 provides an insight into the JavaScript implementation, its structure, and the reasoning behind its design choices. Afterwards, Section 5 presents the results of my replication and discusses the validation of my model's ability to reproduce results that are similar to these of the original paper. Lastly, Section 6 concludes.

2 Related Work

The related work of this thesis deals with the question whether HF traders are beneficial, neutral or detrimental for the market. One might argue that HF traders take over a market-making function, providing liquidity and making market prices more efficient (Carrion 2013, Brogaard 2010). Furthermore, Hasbrouck & Saar (2013) and Jovanovic & Menkveld (2016) find in their empirical investigation that the presence of HF trading causes transaction costs and bid-ask-spreads to decrease, and Brogaard (2010) finds that market volatility is reduced when HF traders participate. On the other hand, there is the concern that there are no legal boundaries for HF traders acting as market-makers, which allows them to exploit their power. Additionally, their status as a liquidity-provider is not reliable (Carrion 2013). The work of Kirilenko et al. (2017) suggests that HF trading also contributes to the emergence of so-called flash crashes, i.e. episodic fragilities where the pressure to sell is high. Furthermore, it is commonly suspected that at least some HF traders use corrosive strategies like front-running[1] or manipulation of the market price, e.g. spoofing (placing orders with the intend to withdraw them before being fulfilled in order to feign increasing or decreasing demand for a stock), herding (appealing to the herd-instinct of other traders and causing them to trade in a certain direction, inducing a feedback-loop) or quote-stuffing (deliberately glutting the market with excess messages) (e.g. Carrion 2013)[2]. As several authors, e.g. Jovanovic & Menkveld (2016) and Xue et al. (2016), have highlighted, speed advantage is the key to success in trading activities because it allows the trader to react to market events and news quicker than their slower competitors. According to Hasbrouck & Saar (2013) nowadays HF trading firms have the tech-

[1] Strikingly, Brogaard (2010) does not find any evidence of abusive front-running activity in his sample data.

[2] For a more thorough overview of the debate about the costs and benefits of HF trading, see the introduction of Jacob Leal et al. (2016).

nology and infrastructure to trade at a latency[3] of 2-3 milliseconds. Only very recently, research has begun on HF traders' activities and their impact on the market. This section gives an overview on a selection of recent scientific studies on high-frequently trading, both empirically and by computational simulations.

2.1 Empirical Investigation in High-Frequency Trading

Hasbrouck & Saar (2013) investigate the impact of HF trading on the market environment. To measure the speed of low-latency trades,[4] the authors use publicly-available NASDAQ data to study message activity triggered by an improvement of the quote. Reactions to this type of event are twofold: an immediate trade as sellers hurry to hit the bid, and competing buyers having a race by canceling and resubmitting bids to stay competitive.

The authors observe a high presence of what they define as "strategic runs", i.e. a "series of submissions, cancellations, and executions that are linked by direction, size, and timing, and which are likely to arise from a single algorithm" (Hasbrouck & Saar 2013, p. 656). These strategic runs can be interpreted as traffic generated by two algorithms strategically responding to each other in a manner that suggests that they attempt to position themselves at the top of the book. The strategic runs found in the data last for only a couple of seconds and can entail hundreds of messages, consisting of submissions and cancellations. Concretely, longer runs of 10 or more messages constitute over 60% of the messages in the sample periods. Moreover, around 35% of the runs are eventually executed, where less than 10% of the runs were terminated with an active placement of a marketable order.

[3] Latency is defined as the time that an algorithm takes to react to an event, calculate its response and send it back to the exchange (Hasbrouck & Saar 2013).

[4] The authors deliberately use the term low-latency instead of high-frequency. According to them, HF trading is a subset of algorithmic trading comprised of proprietary algorithms that require low latency. On the other hand, low-latency trading can additionally encompass activity of agency algorithms.

Furthermore, Hasbrouck & Saar (2013) analyze the influence of low-latency traders on certain market quality indicators. They find that low-latency trading activity is negatively correlated with the quoted spread, with the total price impact of trades, and with short-term volatility. Furthermore, it is positively correlated with depth in the limit order book (LOB), i.e. the time-weighted average number of visible shares in the LOB for the best posted prices. This leads to their conclusion that low-latency activity creates positive externalities in the market.

In the same journal issue, Carrion (2013) also examines the impact of HF trading on market quality in a NASDAQ dataset, where the trades of HF traders are explicitly flagged as such. The author pursues the question whether HF traders provide liquidity and make prices more efficient or whether their strategies have the opposite effect. Just as Hasbrouck & Saar (2013), Carrion (2013) finds a positive association of high HF trading activity and market quality measures. Concretely, it takes a shorter amount of time for prices to integrate information from order flow and market index returns, implying that HF traders make prices more efficient. Furthermore, HF traders engage in successful intraday market timing, i.e. they buy stocks at temporarily low prices and succeed in selling them at intraday heights, indicating the existence of a certain amount of predictability in intraday prices. Lastly, spreads are wider when HF traders provide liquidity and they are tighter when HF traders take liquidity, wherefore the author concludes: "[HF traders] provide liquidity when it is scarce and consume liquidity when plentiful" (Carrion 2013, p. 681).

The events of May 6, 2010, known as the "Flash Crash", are subject to examination in several reports and papers on HF trading, such as Paddrik et al. (2011) and CFTC-SEC (2010). This crash was initiated by an algorithmic trader who sold 45,000 contracts in 20 minutes. Initially, the sell pressure was absorbed by HF traders and other intermediaries, until fundamental traders and market makers

dropped out of the market, leaving HF traders to trade among themselves. This drove the price down even further and resulted in a buy-side depth of less than 1% of what was usually observed. Trading volume and the number of trades on that day were more than double compared to the prior days. Volatility was significantly larger as well. Due to the interconnectedness of markets, other securities' prices plunged as well, some of them up to 40%. The market has crashed endogenously, without the occurrence of a fundamental shock (CFTC-SEC 2010, Paddrik et al. 2011).

A widely cited paper on HF trading during the Flash Crash is Kirilenko et al. (2017), which examines 15,000 trading accounts in the E-mini S&P 500 stock index futures market. HF traders are identified by applying criteria such as the accumulation of only small directional positions, a high participation rate, a high cancellation rate and their inventories that display mean-reversion by the end of the day. The findings include that HF traders, who can react faster to a signal, are able to exploit stale price quotes. Regarding the Flash Crash, HF traders account for most of the aggressive sell volume and contrarily to Carrion (2013) and Hasbrouck & Saar (2013), Kirilenko et al. (2017) stress the vulnerability of financial markets due to HF trading and warn that HF traders can significantly contribute to the emergence of further flash crashes.

Cohen & Szpruch (2012) use the findings and the description of HF traders' activity in Kirilenko et al. (2017) to build a theoretical model of the interactions of fast and slow traders. They focus on the HF traders' "predatory" trades using front-running strategies, along which they have no net position before or after the trades are complete, also known as "round-trip". This strategy allows HF traders, who have a high temporal risk aversion, to minimize the exposure to price risk (Kirilenko et al. 2017). In Cohen & Szpruch (2012)'s model, the HF trader is able to make a risk-free profit – or what they call "latency arbitrage" – using the round-trip

strategy. Nevertheless, this profit is not unlimited, as the HF traders can only exploit the volume LF traders are willing to trade. The LF trader sustains a loss due to the presence of HF traders. Since the HF trader's order arrives at the stock market faster, the LF trader's order (of the same stock and the same direction as the HF trader's order) is executed further along the limit order book. This effect is called "slippage". Still, in the model, when the LF trader becomes aware of the presence of the HF trader, the LF trader keeps participating, but with a smaller volume, leading the authors to the conclusion: "Therefore, [the HF trader's] presence does not render the market unworkable, even though it does introduce a deadweight loss in the market." (Cohen & Szpruch 2012, p. 213)

2.2 Artificial Stock Markets and Agent-Based Models

The formation of stock prices in financial markets can be attributed to a range of factors. These dependencies are difficult or even impossible to detect in empirical studies, as these studies cannot isolate the pure contribution of a factor to occurrences in the stock market. Furthermore, empirical studies fail to give answers to scenarios that have never occurred before, e.g. effects of changes in specific market regulations or effects of extreme market situations. Therefore, researchers make use of agent-based models of financial markets, in which the activity of traders are simulated. These artificial markets allow to assess the influence of single factors to market quality indicators and to run thought-experiments to various scenarios for which no or little historic data exists (Mizuta 2016).

An agent-based financial market is a so-called "bottom-up model", i.e. only the micro level is explicitly modeled, like the agents or the financial exchanges. These agents are modeled to have individual behavior and interaction strategies, so that macro phenomena, such as price variations, can be observed as a result of their interplay. Features on the macro level are not modeled explicitly; they emerge as

an outcome of the simulation (LeBaron 2002, Mizuta 2016).
At this point it is important to note that one should still exercise prudence when simulating a model. As (Mizuta 2016, p. 1-2) points out, artificial market simulations ought to "show possible mechanisms affecting price formation through many runs and gain new knowledge; conversely, a limitation of artificial market simulation is that their outputs may, but not certainly, occur in actual financial markets". Indeed, a market model should be general enough to produce outcomes that are probable in actual markets. A model that is simple helps to avoid the over-fitting problem. According to (Chen et al. 2012), too complex models are often criticized for being too difficult to evaluate. The risk is that in complex models, arbitrary results are produced by over-fitting too many parameters (Mizuta 2016).

In his paper "Building the Santa Fe Artificial Stock Market", LeBaron (2002) describes his observations and findings from his work on one of the first agent-based financial market models, built in the early 1990's. The Santa Fe Artificial Stock Market – or "SFI market", as he calls it in his paper – is a quite simple model, with homogeneous agents using a matching rule for the current market conditions to decide whether to sell or buy one asset in every time period. The price is given endogenously reflecting either excess demand or supply. There's two assets: a risk-free bond, paying a fixed interest rate, and a risky stock, paying a dividend following a stochastic autoregressive process. The agents use a system of classifiers to base their choices on, namely in calculating expected values of the first and second moments for stock returns. These classifier rules map the conditions into forecast parameters. Furthermore, the SFI market includes a genetic algorithm, enabling the agents to learn over time. The probability of activation of the genetic algorithm determines the learning speed of the agent. While discussing the power as well as the shortcomings of the way the SFI market is designed, the author states

that the model manages to replicate several features similar to actual financial data, e.g. return series having an excess kurtosis, very little linear autocorrelation, and constant volatility. Nevertheless, the author warns that the sensitivity of results in the SFI market to learning speeds could be a potential weakness. In his simulations, a single genetic algorithm parameter immensely changes the outcome, wherefore he advices researchers to keep in mind that one can never be certain concerning potential model predictions.

Recently, not only researchers in the academic field are increasingly interested in agent-based models of HF trading, but also financial regulators and stock exchanges (Mizuta 2016). The goal is to gain insights in the effects of e.g. market regulation changes (Jacob Leal & Napoletano 2017), transaction taxes (Pellizzari & Westerhoff 2009), circuit breakers (Kobayashi & Hashimoto 2011) or order cancellation fees (Veryzhenko et al. 2016).

Jacob Leal et al. (2016) investigate the interplay between LF and HF traders and the effects on asset price dynamics. The authors construct a model, in which LF agents trade against HF agents, whereby HF agents have perfect information regarding the offer submissions by the LF agents. This way, HF agents can exploit and profit from the information asymmetry. The findings include that the presence of HF traders increase market volatility and that it plays an essential role in the generation of flash crashes. An in-depth description of the ABM and the results are given in Section 3.

This ABM also serves as a model for the work of Xiong et al. (2015). However, they adjust the ABM such that HF traders act as market makers, i.e. they provide liquidity to other market participants and make profits by gaining the bid-ask spread. The key difference to the ABM in Jacob Leal et al. (2016) is that HF agents submit both a sell and a buy order (instead of only one order with a random direction of equal probability) in order to absorb the orders of the LF

agents and earn the profit on the spread. The authors use the simulations to study HF agents' market making strategies. These strategies include e.g. placing orders at the current best bid and ask prices, such that the profit exactly equals the spread in the LOB, or place the order deeper into the LOB in order to gain larger profits, but at the risk of not having the order executed. Another strategy is to reduce the volume of the inventory when market price volatility increases, which is especially useful when the market price does not evolve according to a random walk, as the findings of Xiong et al. (2015) suggest. The authors also research the impact of various levels of competition amongst market makers. While in the base-case simulation, the ratio of HF agents to the total number of agents is 2%, in the competition-scenarios the number of HF agents is increased as well as trading latencies are arranged to vary. The authors find that trading speed is essential in the simulations. Faster HF agents are likely to earn much more than slower ones, while with an increasing number of HF agents the returns are zero, and even negative when losing the competition.

Jacob Leal & Napoletano (2017) also use the ABM from Jacob Leal et al. (2016) to make extensions in order to study the effectiveness of regulatory policy changes directed towards HF traders on market volatility and on the occurrence and the duration of flash crashes. The ABM is constructed in the same way as in Jacob Leal et al. (2016), but some features are added. HF traders are able to withdraw their order if their estimated return is negative, mimicking the HF traders' ability to quickly cancel their orders. Policies that aim to reduce the cancellation rate like minimum resting times or cancellation fees come off well in the Monte Carlo simulation runs. Both policy measures dampen market volatility, as they slow down HF traders. This way, HF traders are blocked from reacting aggressively on market news. However, while the number of flash crashes in the simulations is reduced, there is a trade-off of a longer duration of these flash crashes. This is due

to the fact that implementing stricter rules on order resting times implies a longer memory effect. Therefore, price recovery is slowed down, voiding the positive role of HF trading found in Jacob Leal et al. (2016). Jacob Leal & Napoletano (2017) also examine other policy measures, i.e. two different variants of circuit breakers (ex-ante and ex-post)[5] and transaction taxes, implemented in stock exchanges in many countries to prevent episodic market instabilities. The effect of high enough transaction taxes is positive regarding market volatility, although its effect is smaller compared to the order cancellation policy. Circuit breakers are designed to halt trading when triggered by a certain price change. While an ex-ante circuit breaker succeeds in preventing the market from generating flash crashes, the introduction of ex-post circuit breakers has almost no effect on market volatility and even a detrimental effect on the duration of flash crashes. The authors conclude that changing regulatory policies can have complex effects on stock markets and remind the reader of the dual role HF traders play: While they often are the source of flash crashes, they help the market to recover faster by quickly restoring liquidity.

[5]The difference between an ex-ante and an ex-post circuit breaker is that the latter is triggered by a relative price change from the last price. While by construction, this cannot stop flash crashes from happening, it can have an impact on its duration. Meanwhile, the ex-ante circuit breaker has a limit up/limit down price, initializing a trading halt before the trading session price is formed. This way, flash crashes can in theory be prevented.

3 Agent-Based Modeling of the High-Frequency Trading Environment

In this section I give a detailed description of the model in Jacob Leal et al. (2016) and my replication of it.

3.1 The Jacob Leal et al. Model

Although only published a year ago, the results and the model in Jacob Leal et al. (2016) are already cited several times in the HF trading literature. In the paper, the authors propose an ABM of the interplay of HF and LF traders, with boundedly-rational[6] HF traders adopting an event-based trading-activation strategy. The aim of the paper is twofold: First, the authors try to assess whether and to what extend HF traders cause periods of high price volatility and second, how HF traders affect occurrences and the length of flash crashes in financial markets. Their results include that the presence of HF trading indeed increases market volatility and that it leads to a higher number of flash crashes. However, HF trading has also a beneficial effect, as it makes prices more efficient and it improves price recovery after a flash crash.

This paper stands out from other literature as it incorporates three features: First, HF traders are modeled in a more sophisticated way than previous literature, allowing HF traders to observe the market in order to exploit the order information of LF traders. Second, the interplay among HF and LF traders is explicitly accounted for. Third, the stock market is able to endogenously generate flash crashes and then recover rapidly (Jacob Leal et al. 2016).

[6]The term "boundedly-rational" is used to describe that the agents are activated based on rules of endogenous events instead of having an exogenously-given trading frequency incorporated. Boundedly-rational agents in ABMs also appear in Brock & Hommes (1998) and Westerhoff (2008), where these agents' decisions and actions follow rational heuristics.

To further investigate in this direction, I build a replication of their model. Furthermore, I propose some improvements and implement them in my model, which I describe in detail in Subsection 3.2.

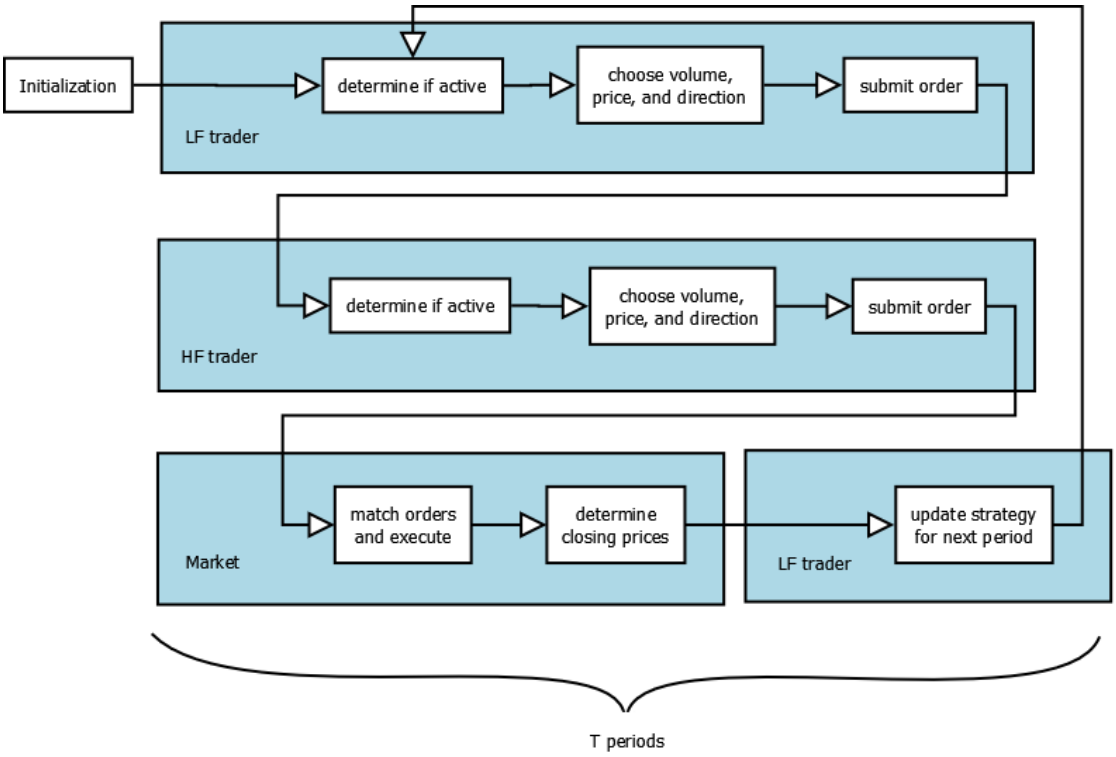
3.1.1 Event Timeline

The ABM is a model of a stock market, in which heterogeneous agents trade one type of asset with each other by submitting buy or sell orders in the LOB. The simulation runs for T periods, where one period is assumed to represent one minute. In each period, each agent has a certain probability to be activated, depending on their type. If active, they can decide to submit either a buy or sell order. The two types of agents are LF traders, following one of two possible strategies, and HF traders, following an event-based strategy. Concretely, the process looks as graphically depicted in Figure 1. The agents which start off are the LF traders. Depending on the strategy they have set in the prior period,[7] they choose price, volume, and direction (i.e. whether to buy or to sell the asset). Afterwards, it is the HF traders' turn. While both types of traders know the past closing price as well as the past and current fundamental values of the asset, HF traders also know the sizes and prices of the orders that have just been submitted by LF traders. All submitted orders by LF and by HF traders must specify the size, direction and limit price. When all orders are submitted in the LOB, the orders are matched, with priority to price, then to arrival time, and then to LF traders. Orders that are not matched stay in the LOB until they expire. After all orders are executed, the closing price and each agent's profit of the period are calculated, where the closing price is the last executed transaction of the current period.[8] In the end,

[7] In the initialization round, each HF agent adopts either strategy with a probability of 0.5 (Platt & Gebbie 2016).

[8] In an earlier version of the paper, the closing price was defined as the maximum of all executed trades of the current period, as is discussed in more detail in Subsection 3.2.2.

Figure 1: Flowchart of the event timeline of one trading session. Own image.



knowing the closing price and the hypothetical profit if having chosen the other strategy, LF traders can choose to either keep their strategy or switch for the following period.

According to Jacob Leal et al. (2016), the reason why HF orders are modeled to be inserted in the LOB after LF orders and before the actual matching takes place is to replicate the way HF traders use their low-latency technology (as described in Hasbrouck & Saar (2013)) to exploit LF traders. HF traders have such a low latency when trading that they can process information released by LF traders first, and have orders arrive at the stock market simultaneously with LF traders.

3.1.2 Description of the Agents

The two types of traders differ in several factors: First, HF traders' orders expire very quickly (mimicking the empirically measured high rate of submitting and canceling orders (see references in Jacob Leal et al. (2016))). In the simulations of Jacob Leal et al. (2016), the order resting time γ^H is set to one period. Second, the HF traders' trading speed is endogenous: The activation rule incorporates a dependency on price fluctuations, namely the agent is activated when the relative price difference of the last and the before-last period is larger than some random threshold Δx_j, individually drawn from a bounded uniform distribution. Therefore, HF agent j participates if:

$$\left| \frac{\bar{P}_{t-1} - \bar{P}_{t-2}}{\bar{P}_{t-2}} \right| > \Delta x_j, \tag{1}$$

where $\bar{P}_t$ is the closing price of the asset at period t.

Third, their strategy is to profit by using their knowledge about the LF agent's trades. The direction of the order to be submitted does not play a role, and thus, if activated, they submit a buy or sell order with a probability $p = 0.5$ each. The volume $D_{j,t}$ of their order depends on volumes available in the opposite side of the LOB.

$$D_{j,t} \sim Poi_{\text{trunc}}(\lambda V), \tag{2}$$

where V is the total volume present in the opposite side of the LOB and $0 < \lambda < 1$ is a weighting parameter. Hence, the volume to be submitted is drawn from a truncated Poisson distribution (bounded by $\pm$ 3000 and $< \frac{1}{4}V$).

HF traders set their limit price according to following formula:

$$\begin{aligned} P_{j,t} &= P_t^{ask}(a + \kappa_j), \\ P_{j,t} &= P_t^{bid}(a - \kappa_j), \\ \kappa_j &\sim U(\kappa_{min}, \kappa_{max}) \end{aligned} \tag{3}$$

where P_t^{ask} and P_t^{bid} are the best ask and the best bid price, respectively. This means that HF agents try to trade near the best LOB prices.

LF agents, however, have a different set of rules regarding order expiration, activation, and strategies. Their orders expire after a longer amount of time γ^L, where $\gamma^L > \gamma^H$. In the parametrization of the authors' model, their orders expire after 20 rounds. Furthermore, contrarily to the event time framing of a HF trader, the LF trader participates in a chronological context, i.e. the trader becomes active exogenously, with the trading frequency being constant over time and drawn from a truncated exponential distribution. Lastly, LF agents follow either a fundamentalist or a chartist strategy. The fundamentalist strategy is based on the fundamental value movement of the asset. As the fundamental value F_t evolves according to following random-walk formula:

$$\begin{aligned} F_t &= F_{t-1}(1 + \delta)(1 + y_t), \\ \delta &> 0, \\ y_t &\overset{iid}{\sim} \mathcal{N}(0, \sigma^y) \end{aligned} \tag{4}$$

the order $D_{i,t}^f$ of LF agent i is determined as follows:

$$\begin{aligned} D_{i,t}^f &= \alpha^f(F_t - \bar{P}_{t-1}) + \varepsilon_t^f, \\ 0 < \alpha^f &< 1, \\ \varepsilon_t^f &\overset{iid}{\sim} \mathcal{N}(0, \sigma^f) \end{aligned} \tag{5}$$

Note that D is a *directed* order, meaning that for sell orders, D is negative, and positive for buy orders. The chartist strategy is based on the movement of the last two closing prices:

$$\begin{aligned} D_{i,t}^{c} &= \alpha^{c}(\bar{P}_{t-1} - \bar{P}_{t-2}) + \varepsilon_{t}^{c}, \\ 0 < \alpha^{c} &< 1, \\ \varepsilon_{t}^{c} &\overset{iid}{\sim} \mathcal{N}(0, \sigma^{c}) \end{aligned} \tag{6}$$

LF agents set their limit order price $z_{i,t}$ ticks away from the previous period's closing price:

$$\begin{aligned} P_{i,t} &= \bar{P}_{t-1}(1+\delta)(1+z_{i,t}), \\ z_{i,t} &\sim \mathcal{N}(0, \sigma^{z}) \end{aligned} \tag{7}$$

After the LF and HF agents' orders are matched and executed, and the closing price $\bar{P}_t$ is determined, both types of agents compute their profits of the current period π_t in the equivalent manner:

$$\pi_{j,t} = (\bar{P}_t - P_{j,t,})D_{j,t} \tag{8}$$

for HF agents and

$$\pi_{i,t}^{st} = (\bar{P}_t - P_{i,t,})D_{i,t}^{st} \tag{9}$$

for LF agents, where $(st = c)$ denotes the chartist and $(st = f)$ the fundamentalist trading strategy.

Before the end of each round, LF traders decide which strategy to adopt in the following period. This decision depends on the relation of the actual profit made in this round to the profit if the other strategy had been adopted. More specifically, the probability to adopt the chartist strategy in the following period ($\Phi_{i,t}^{c}$) is

determined by:

$$\begin{aligned} \Phi_{i,t}^{c} &= \frac{e^{\pi_{i,t}^{c}/\zeta}}{e^{\pi_{i,t}^{c}/\zeta} + e^{\pi_{i,t}^{f}/\zeta}}, \\ \Phi_{i,t}^{f} &= 1 - \Phi_{i,t}^{c}, \end{aligned} \tag{10}$$

where $\zeta > 0$ is the switching intensity parameter.

This method of determining the weights for the strategy has been employed by several authors (e.g. Westerhoff (2008) and Pellizzari & Westerhoff (2009)), originally proposed by Brock & Hommes (1998). This method suggests that agents switch based on the profitability of the two strategies, whereby ζ indicates the agent's elasticity. For $\zeta \to \infty$, Φ approaches $\frac{1}{2}$, implying that the agent is indifferent between the two strategies and adopts one of them with equal probability. Contrarily, for smaller ζ, the agent becomes more elastic and inclined to switch to the more profitable strategy for the following round (Pellizzari & Westerhoff 2009).

3.1.3 Results of Jacob Leal et al. (2016)

When running Monte Carlo simulations first with and then without HF traders, volatility of price returns proves to be significantly higher in the first case, stressing the destabilizing role of HF trading. Furthermore, there are several flash crashes when HF traders are present in the market model, while there are none when they are absent (where a flash crash is defined as a price drop of at least 5% with a sudden recovery within 30 minutes). Flash crashes emerge in the simulations when three events occur: a high bid-ask spread, a strong concentration of LF traders on the buy side, and the synchronization of HF traders on the sell side of the LOB. Nevertheless, in the model, HF traders also exhibit beneficial effects: First, the relative difference between price and fundamental value of the asset is significantly

lower when HF traders are present, suggesting that they ameliorate price efficiency. Second, the trading-to-book-volume ratio is higher with HF trading. Third, HF traders are able to react quickly to market information and to cancel obsolete orders quickly from the market. This helps recovering prices after a price downturn. As a conclusion, the authors say that there is a "trade-off between volatility and the incidence of extreme events, on one hand, and price-resilience, on the other hand" (Jacob Leal et al. 2016, p. 69).

3.2 Critical Assessment and Amendments to the Model

Jacob Leal et al. (2016) succeed in building a model that is able to replicate the main properties of financial markets: e.g. the autocorrelation values of price returns, which do not reveal any patterns and which are almost zero, the presence of volatility clustering, and the existence of fat tails. This suggests that the model is indeed based upon sound principles (Mandelbrot 1963, Cont 2001). Nevertheless, some model design choices merit further investigation.

3.2.1 Profit Calculation

At the end of each period, the agents' profits are calculated as shown in Equations 8 and 9, i.e.

$$\text{Profit} = (\text{Closing Price} - \text{Individual Trading Price}) * \text{Offered Quantity}.$$

It does make sense to calculate the difference between the closing price and the individual trading price. This way, one can determine whether the choice of the individual trading price has caused gains or losses for the current period. Then, according to Jacob Leal et al. (2016), this difference is multiplied with the offered quantity. To my understanding, this calculation yields the return given that the

offered quantity gets matched and fully executed. It would make more sense to instead use the actually traded quantity in the calculation. This way, one would not obtain the expected profit conditional on a full execution, but the profit that has actually been made this period.

However, the profit calculation is used by LF agents to determine their strategies for the following period (by comparing the actual profit to the profit if the other strategy had been adopted). However, it is impossible to know how much of the submitted order would actually have been executed given the LF agent had chosen the other strategy. Therefore, I add an extension in which the offered quantity is multiplied by a weight between zero and one to account for the fraction of the submitted order actually being traded. Due to the lack of data on the true distribution of this fraction, I simply use a uniform distribution. Of course, this assumption is not founded on any empirical observations; it is merely a means to find out whether multiplying the volume with a random variable significantly changes the outcome, and if yes, to what extent. The results are presented in Subsection 5.3.

3.2.2 Determining Closing Prices

After the LF's and HF's marketable orders are executed, the closing price is calculated, which is defined to be the last executed transaction of the current period. In an earlier version of Jacob Leal et al. (2016), the closing price was defined as the highest price of all executed trades of the current period. In the newest version, the authors claim that the results do not differ when using the maximum price, the last price or the average price as the closing price. Yet it does seem unintuitive to use maximum prices as closing prices. First, it introduces an asymmetry. Due to how profit calculation is defined, profits are never negative when using this method. The profit formula contains the difference of the closing price and the

individual trading price. If the closing price is always the maximum price, this difference is always either positive or zero. This is hardly a depiction of a real market. Second, it is hard to justify that prices always increase towards the end of a time period, and even reliably reach its peak. Furthermore, this model design choice seems to be somewhat arbitrary, as the same arguments that speak for the maximum price as the closing price can also speak for the minimum price as the closing price.

In an attempt to investigate the effect on the results of the different ways of determining closing prices, I implement five different methods: last price (the default version), maximum price, mean price, minimum price and random price as the closing price, respectively. The results are presented in Subsection 5.3.

3.2.3 Number of Trades per Round

In every round, both LF and HF traders each submit at most one single trade offer. This choice of implementation might seem questionable on first sight. In this paper, HF traders are explicitly designed to exploit the order information released by LF traders for their own profits. However, it could be even more profitable for HF traders if they were allowed to make two trades in one time period. More specifically, HF traders could simultaneously release a counter-trade in order to collect the spread (similar to the ABM of Xiong et al. (2015)). Without making this counter-trade they risk an asset price movement to their disadvantage. This strategy would be consistent with actual financial markets. According to Hagströmer & Norden (2013), between 63% to 72% of the HF trading volume can be accounted to market makers, i.e. traders who submit ask *and* bid offers, and thus gain the spread. Moreover, according to Carrion (2013, p. 683), HF traders are "reluctant to hold inventory overnight" because of said inventory risk, and therefore, they are inclined to keep low net positions compared to their high

trading volume during the day (Kirilenko et al. 2017).

Nevertheless, due to how the model is designed, one could argue that this very behavior is already implemented. Since LF traders have such a low trading frequency in this simulation configuration, the net effect is as if HF traders would trade a few times consecutively.

3.2.4 Batch Matching

Although "batch matching" is quite standard practice in several papers (e.g. Maslov 2000, Preis et al. 2006), it is evident that it is not realistic. "Batch matching" is the mechanism of first having all orders arrive at the market and then processing the whole batch simultaneously, which has also been criticized in Platt & Gebbie (2016). In a real stock market, orders arrive and are executed on a continuous basis, instead of at the end of a specific time period. In order to investigate whether batch matching allows a close enough representation of a real market, I build in a switch in my model that allows to change the matching mechanism to a continuous market, where orders are directly matched upon arrival. I comment on the results in Subsection 5.3.

3.2.5 Order types

In the model specification, agents are only allowed to submit limit orders, i.e. they must designate a price. This design choice is justified given that Hagströmer & Norden (2013) find that up to 86% of the orders in their sample data were limit orders. In order to account for the remaining 14%, it would be interesting to find out whether a financial market simulation, where other order types are allowed (like e.g. market orders), would ameliorate the replication. As this modification would require a whole new set of strategy definitions for the agents, it would go beyond the scope of this thesis and is therefore not implemented.

3.2.6 Changes by Platt and Gebbie (2016)

Platt & Gebbie (2016) explicitly replicate the ABM of Jacob Leal et al. (2016). They implement two modifications, which however do not change the results, as they claim. First, they invent their own method of initialization, as Jacob Leal et al. (2016) do not reveal how the model is initialized. In the first round, *all* LF traders participate in the market and place an order according to one of the two strategies, which is determined randomly, with equal probability. They justify their choice of having all LF traders participate in the first round by citing empirical papers that show that at the beginning of the trading day, there is often a peak in trading activity. I therefore apply this initialization method to my model as well. Second, by the way Jacob Leal et al. (2016) choose the parametrization (see Tables 1, 2, and 3), it happens that the LF agent typically submits order sizes smaller than 1. While this is already somewhat odd, as in real markets order sizes smaller than 1 do not exist, the real problem is that HF traders' order sizes depend on the LF traders' order sizes. Jacob Leal et al. (2016) use a Poisson distribution to set the HF traders' order sizes, which causes those to be close to 0 most of the time. Thus, Platt & Gebbie (2016) use an exponential distribution, which makes the HF traders' order sizes bigger.

I implement the simulation in such a way that the HF traders' strategy of setting the order volume is easily switchable between a truncated Poisson and a truncated exponential distribution. Although the traders' order sizes are indeed larger when using an exponential distribution, the model also works and produces the same results when using the setting as in Jacob Leal et al. (2016). Therefore, I keep the default setting to choose HF trader order sizes according to a truncated Poisson distribution.

4 Robust JavaScript Implementation

For the implementation of the ABM I use the programming language JavaScript. JavaScript is a dynamically typed[9] scripting language with prototypal inheritance[10] and structural subtyping.[11] Among others, it offers first class functions as well as array and object literals.

JavaScript was initially intended to be interpreted by a web browser exposing the Document Object Model (DOM)[12] to the script to create dynamic HTML pages. However, JavaScript is in its own right a powerful language with object-oriented, functional and structural elements. `Nodejs` is an open-source, server-side JavaScript interpreter which extends the language with a very basic version of modules (e.g. require) and the usual server-side capabilities such as the filesystem access (Nodejs 2017).

However, JavaScript is also a language that was designed to be easy to use and to be expressive. Consequently, there are many rather unsafe features such as type coersion[13] or implicit global variables[14]. It is therefore standard practice to only use a subset of the JavaScript language without the features that unintentionally behave self-destructively (colloquially: 'foot-guns'). For that purpose I use `jshint`, a JavaScript code quality tool that helps to detect errors and potential problems (JSHint 2017).

[9] The type of a variable can change at run-time.

[10] Prototypal inheritance means that new objects can inherit the properties of old objects (instead of inheriting from classes when using classical inheritance) (Crockford 2008).

[11] Structural subtyping means that any two objects that have the same properties can be passed to the same functions.

[12] The DOM is a programming interface for HTML and XML documents constructed as a tree of objects. It provides a connection between web pages and scripts or programming languages (Network 2017).

[13] Wherever possible, JavaScript tries to convert a variable to the needed type, e.g.: `"0" == false`.

[14] If a variable is assigned without defining it first, JavaScript treats it as a global variable.

4.1 Preliminaries

There are several goals when designing code: Correctness, Modularity, and Readability. The following sections offer a brief overview of my understandings of the works of Crockford (2008), Daggett (2013), Steyer (2014), Osmani (2012).

4.1.1 Correctness

Correctness means that code does exactly what it promises to do and keeps doing so even when extensions are added. In order to ensure this and to raise confidence of code correctness, I adopt measures like e.g. asserts, type-checking, and unit tests.

4.1.2 Modularity

Modularity means that individual objects are decoupled. This way, one can swap and exchange modules, as these modules only depend on each other over interfaces. Modularity in object-oriented languages is achieved by building the logic around objects (or classes) which ideally adhere to the SOLID-Principles[15]. This requires subtyping which in JavaScript is handled out of the box thanks to the structural subtyping. One could now either build a hierarchy of classes, build a prototype chain or use composition over inheritance and make the code completely inheritance free.

[15]The term SOLID stands for the basic principles of programming: single responsibility, open-closed, Liskov substitution, interface segregation and dependency inversion. Single responsibility means that a module should only do one thing and not be responsible for the performance of several tasks. The open-closed principle says that you should not need to change code inside of a module to extend it. The Liskov substitution is the principle that if a module works with a type, it must also be able to work with a subtype. Interface segregation means that a module should only expect of its inputs that which it really needs in order to work. Lastly, dependency inversion refers to the principle that a module should be given everything it requires to work, e.g. an `OrderBook` does not create a `Clock`, but it is given one. These principles can be seen as guidelines to ensure correctness and modularity and to avoid readability problems.

This way, when implementing new features, old modules do not need to be touched, which reduces the risk of breaking the code.

4.1.3 Readability

Readability means that code should be written in a way that humans can understand what happens just by reading it. The computer does not particularly care how the code is written, but it is important that any person who works or will work with the code in the future exactly understands what is happening. This way, bugs can easier be detected or even not introduced at all. Readability can be enhanced when ensuring that function names are precise, e.g. contain a verb to describe what the function *does*. Also, the output has to be obvious. For example, for the function `areOrdersAutoExecuted`, the only logical return argument is a boolean, i.e. `true` or `false`. Everything else would be an unexpected answer and should thus be avoided. The same applies to variable names. A variable called `numberOfTraders` is expected to be a non-negative number. Other measures are keeping a logical outline, trying to avoid too much nesting and meaningful function signatures. In a nutshell, this standard is called the "principle of least surprise".

4.2 Design

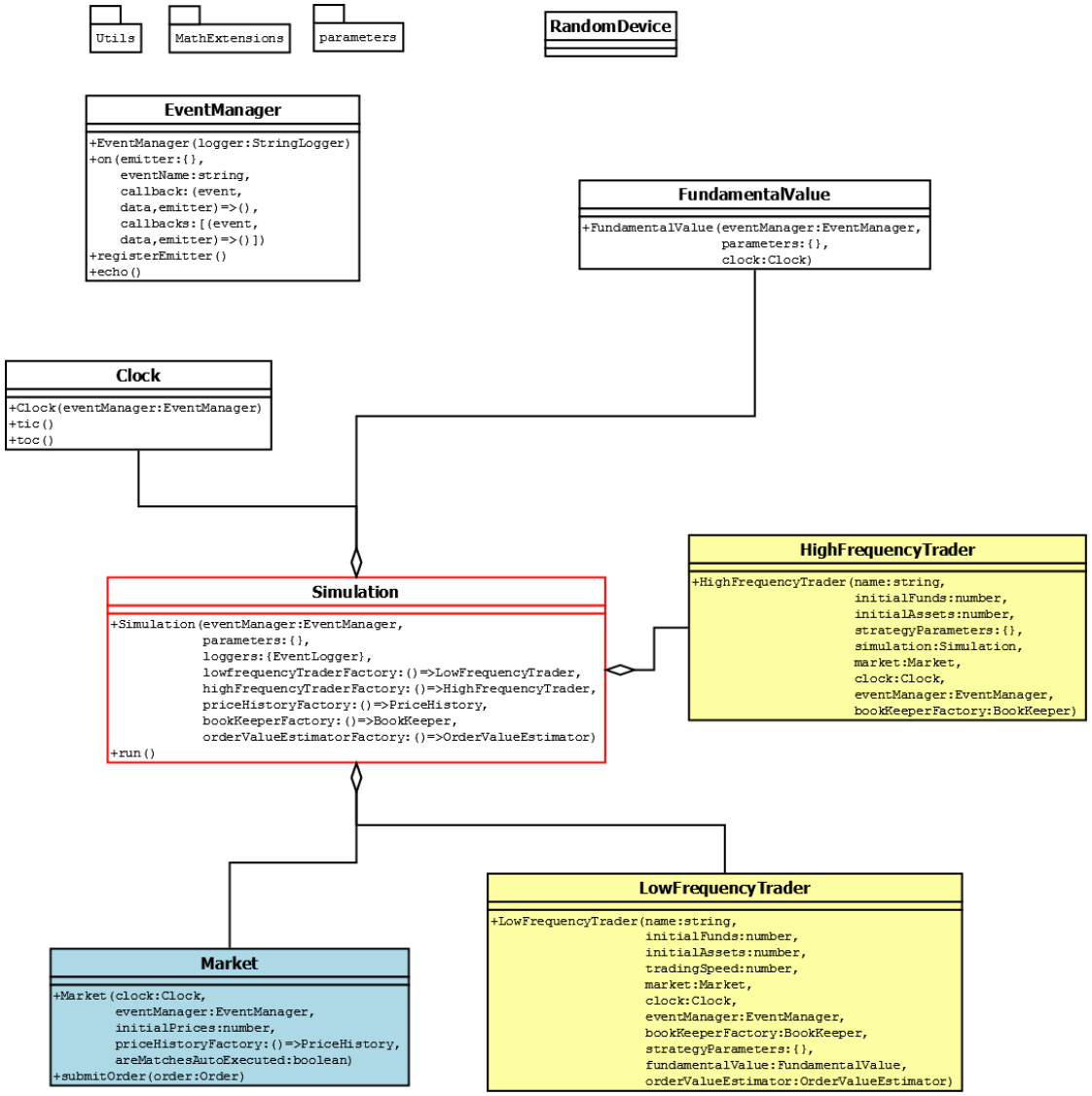
4.2.1 Factories

The implementation builds on several factories[16], which each introduce a type. In order to visualize the structure and the connections of the types to each other, I illustrate the implementation in a UML (Unified Modeling Language) Diagram. The core that runs the simulation is of the *Simulation*, as depicted in Figure 2. It is connected to several other type objects, as indicated by the "has a" arrows.

[16] A factory is a function which creates an object.

Specifically, it has a *Clock*, which manages the start and end of every time period, a *FundamentalValue*, which calculates the fundamental value of the asset in every period, a *Market*, which handles orders, as well as the *HighFrequencyTraders* and *LowFrequencyTraders*.

Figure 2: Core types.



A *Market*, in turn, has the task to submit orders in the *OrderBook* and to keep track of the closing prices (Figure 3). There are several possibilities to calculate the closing price each of them being a subtype of *PriceHistory*.

Figure 3: Market and Order types.

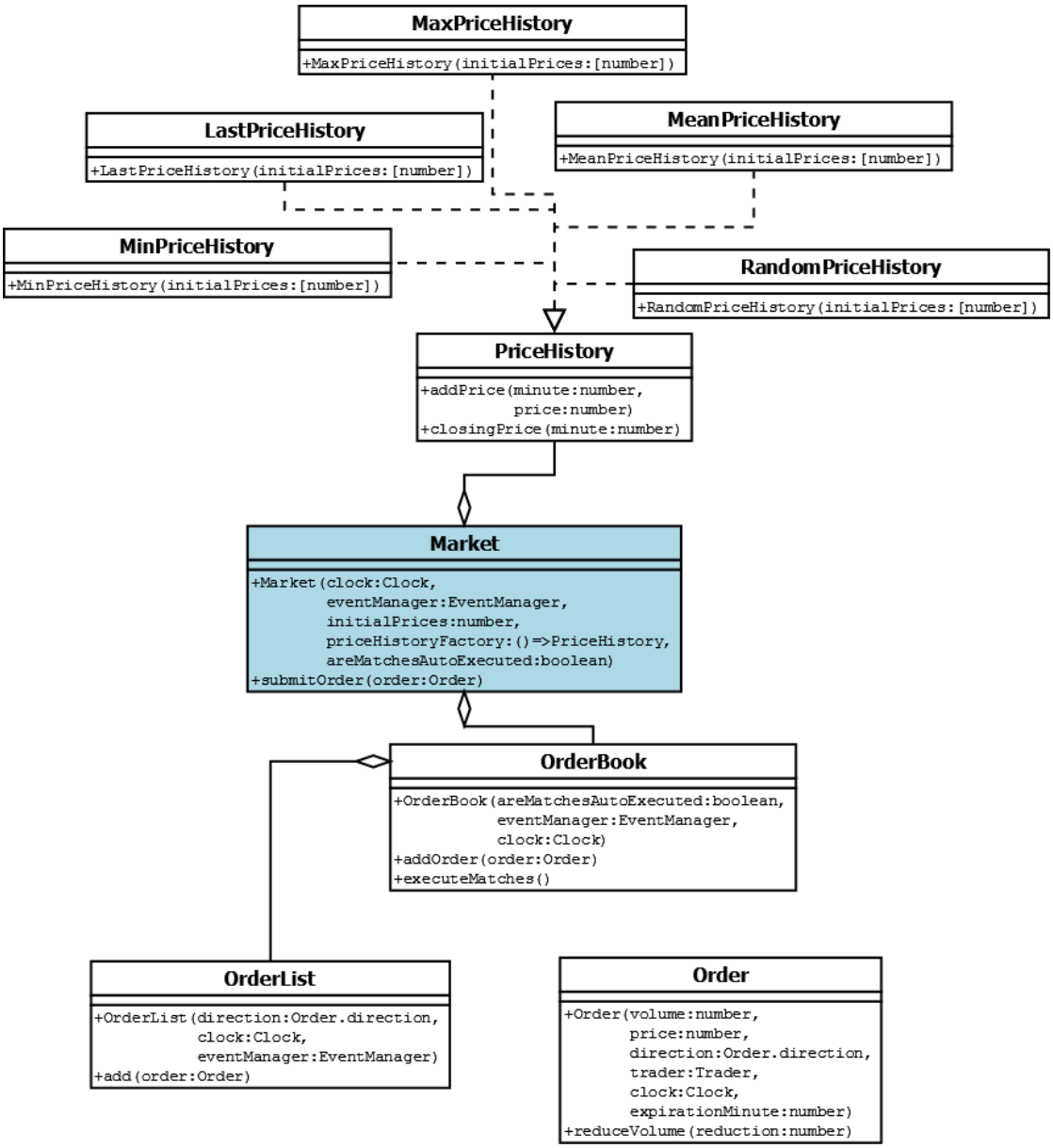


Figure 4 revisits the two trader types which are connected to the *Simulation*. Both traders have their own *Strategy*. The LF trader *Strategy* is more complex than the HF trader *Strategy*, as the LF trader has two different plans of actions to choose from, which are both more computationally demanding. The UML diagram of the LF *Strategy* continues in Figure 5.

Figure 4: Trader types.

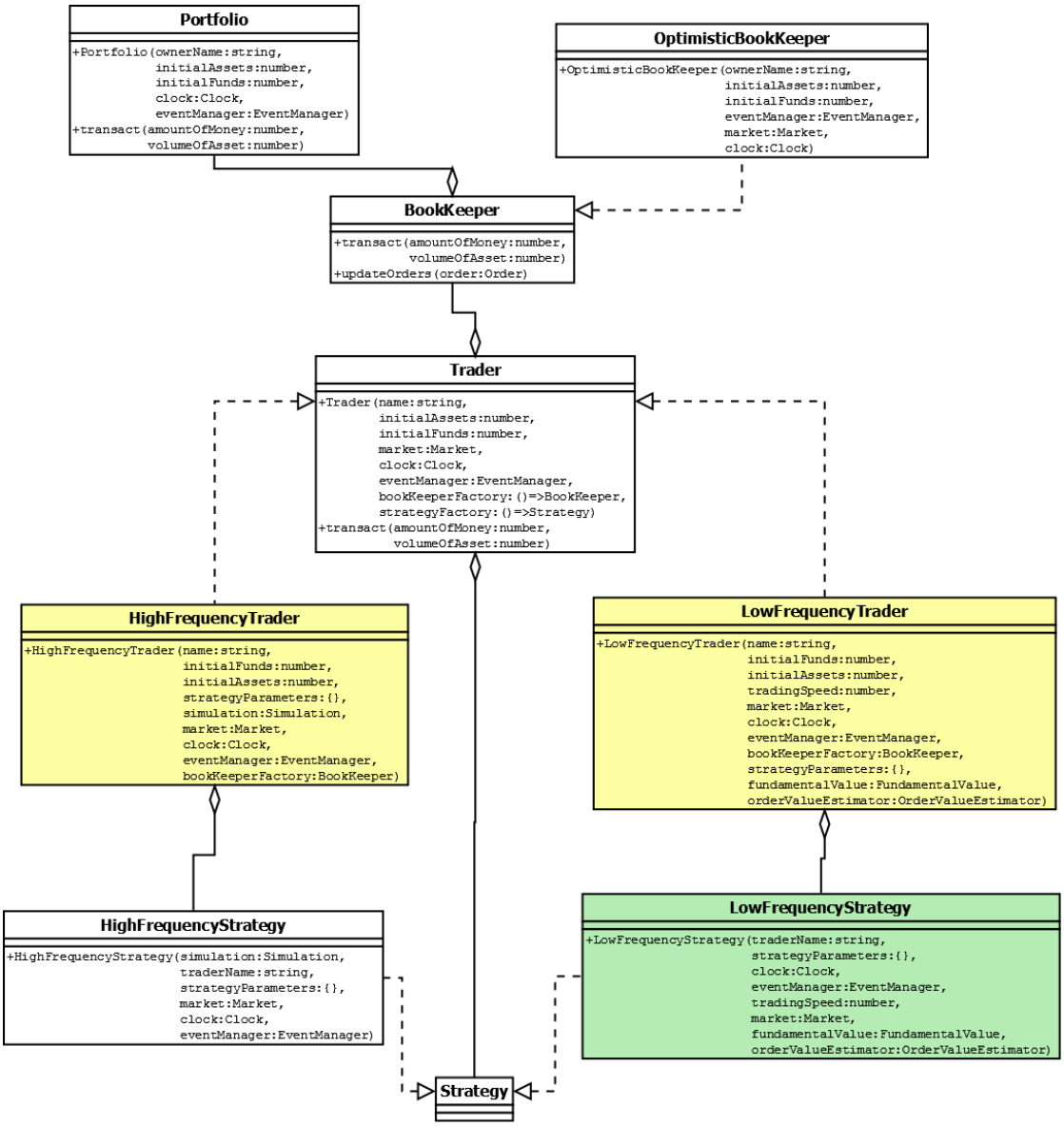
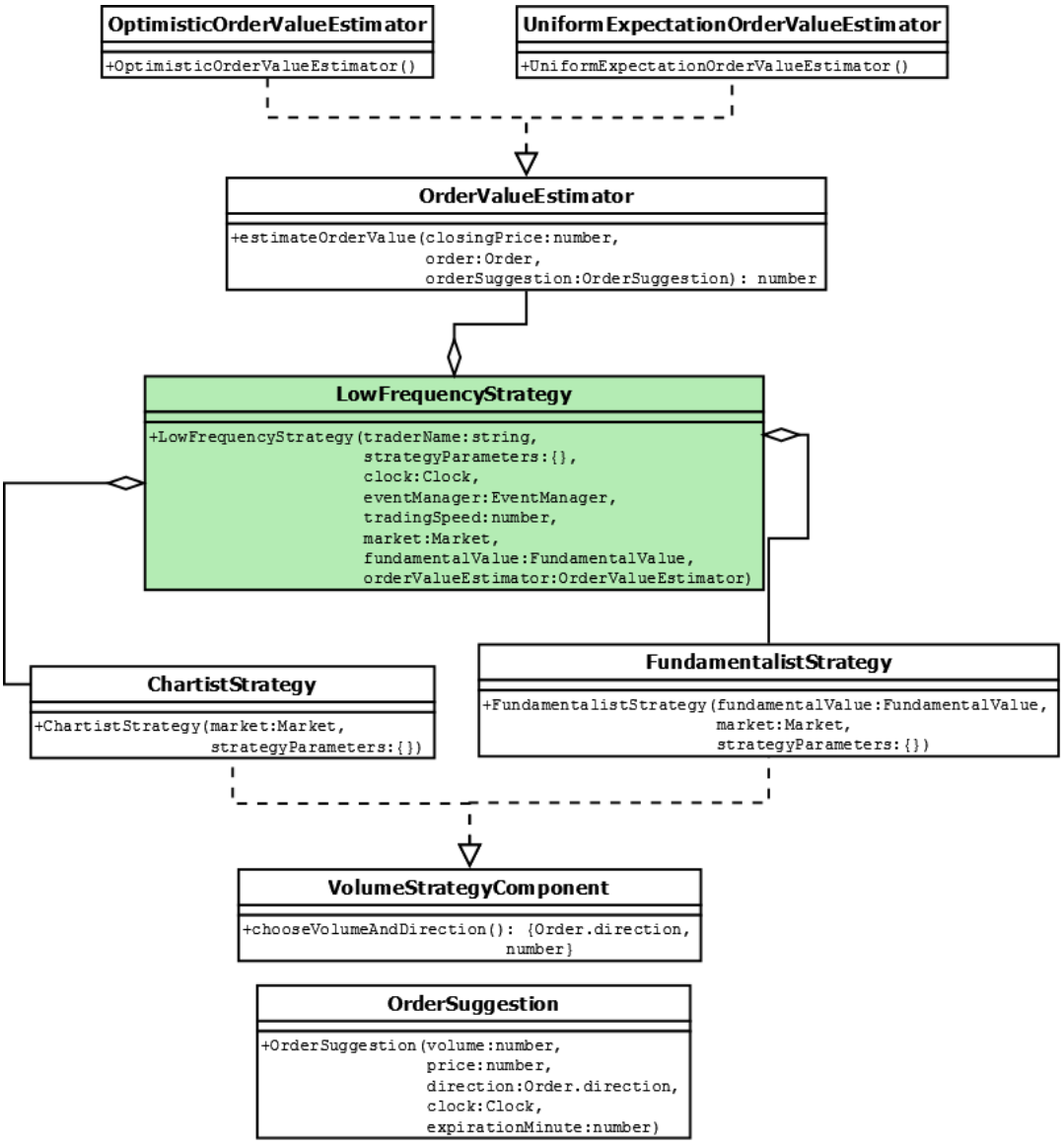
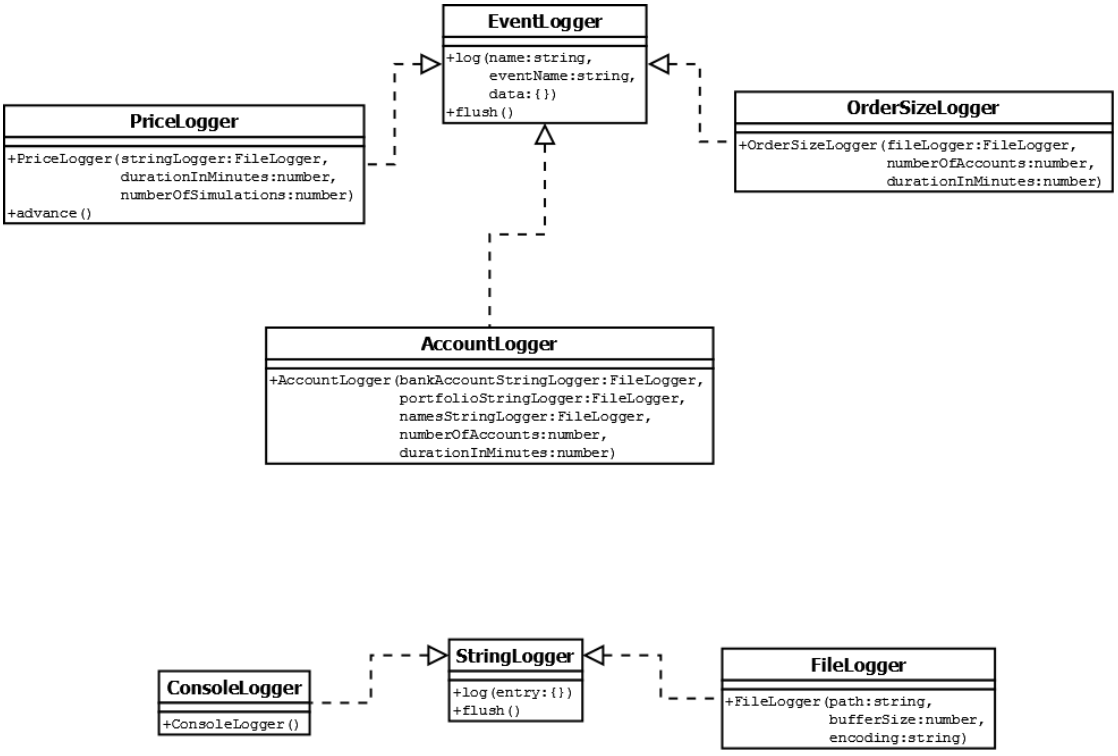


Figure 5: LF Trader Strategy.



The logger types are independent of the other types (Figure 6). The price, account and order size loggers are all an *EventLogger*, i.e. they listen for an event to happen and log the event information as a string in the memory. A *StringLogger* logs a string from the memory onto the console or into a file.

Figure 6: Logging types.



4.2.2 Factory Structure

Every factory is set up in a similar way. In order to facilitate comprehension of my explanations, I present some code snippets, taken from the *Simulation* module. Each module begins with the import of the factories and objects that the module depends on.

```
1 const Utils = require("./Utils");
2 const MathExtensions = require("./MathExtensions");
3 const Clock = require("./Clock");
```

Listing 1: Import of dependencies.

For some factory imports, there exist circular dependencies, i.e. here, the *Simulation* factory needs the *Trader* and *Market* factories, and vice versa. Therefore, the latter are solely *declared* in the beginning, and *imported* just after the definition of the *Simulation* function.

```
1  let Trader;
2  let Market;
3
4  module.exports = (function() {
5    ... //some code
6    return Object.freeze(Simulation);
7  })();
8
9  Trader = require("./Trader");
10 Market = require("./Market");
```

Listing 2: Dealing with circular dependencies.

Inside the `module.exports` closure, the arguments to be passed to the `Simulation` function are defined. Then it is ascertained that first, the imports have worked, and second, the arguments are provided properly.

```
1  const Simulation = function({eventManager, parameters, loggers}) {
2    assert(() => typeof MathExtensions === "object");
3    assert(() => typeof Clock === "function");
4    assert(() => typeof Trader === "function");
5    assert(() => typeof Market === "function");
6
7    assert(() => implementsInterface(eventManager, "EventManager"));
8    assert(() => typeof parameters === "object");
9    assert(() => typeof loggers === "object");
10
11   ... //some code
12
13 };
```

Listing 3: Assertions.

Afterwards, I define all operations of the respective type. Here, the function `run` is an operation in *Simulation*. All operations are inserted into a `that` object. This object is returned when the *Simulation* factory is called. Furthermore, the name of the factory is returned inside the `that` object as a string as well, in order to support the `implementsInterface` functionality when in debug mode. This facilitates debugging by far because it makes JavaScript a bit "type-safe".

```
const Simulation = function({eventManager, parameters, loggers}) {
  ... //some code

  const run = function() {
    while(clock.minute < tradingTime) {
      console.log(`Simulation running @ ${clock.minute}`);
      clock.tic();
    }
  };

  that = Object.freeze({
    DEBUG_interfaces: ["Simulation"],
    toString() {
      return "simulation";
    },
    run,
  });

return that;
};
```

Listing 4: Inside the Simulation factory.

Moreover, a large part of the implementation operates with event listeners. In the following code snippet, I present an example of an event listener. The function `onLowFrequencyTraderDone` is called as soon as a LF trader has finished submitting an order. If the last LF trader is done, the event with the name `ALL_LOW_FREQUENCY_TRADERS_DONE` is emitted.

```
1  const onLowFrequencyTraderDone = function({emitter}) {
2    assert(() => implementsInterface(emitter,"LowFrequencyTrader"));
3    doneLowFrequencyTraders.set(emitter.name, true);
4    const areAllLowFrequencyTradersDone =
5      doneLowFrequencyTraders.size === numberOfLowFrequencyTraders;
6    if (areAllLowFrequencyTradersDone) {
7      emit({
8        eventName: Simulation.ALL_LOW_FREQUENCY_TRADERS_DONE,
9        onEndOfIssuing() {
10         doneLowFrequencyTraders.clear();
11       }
12     });
13   }
14   return areAllLowFrequencyTradersDone;
15 };
```

Listing 5: Events and Event Listeners.

Now, this event is heard by each *HighFrequencyTrader*, and a callback function is called (`onLowFrequencyTradersDone`). This provokes the start of the HF trader's turn.

4.2.3 Math Extensions and Unit Tests

Since JavaScript does not handle the distributions I need natively, a random number library is required. I write my own to avoid third-party dependencies. This is a strong advantage, as public libraries may contain errors or there might be prob-

lems with licensing. I implemented the mathematical extensions by taking tried, tested, and actively used standard algorithms, such as inverse transform sampling. I tested them for correctness by means of unit tests.

Furthermore, the simulation is reproducible because I produced my own random numbers. This way, I always used the same seed for my simulations. Although it is usually not possible to always set the same seed in JavaScript, it can be made possible by producing own random numbers and then always use them.

5 Results, Validation and Extensions

In order to give evidence of the validity of my replication of the model I insert the same default parameters as in the original model. These parameters are described in detail in Tables 1, 2, and 3. Some parameters are not provided in Jacob Leal et al. (2016), hence some assumptions about them are made, listed in Table 4. Using these parameters my replication of the model is able to reproduce the same stylized facts of the return time series as in Jacob Leal et al. (2016) and in Platt & Gebbie (2016), which will be presented in detail in Subsection 5.2.

Table 1: Parameter values of the LF trader as given in Jacob Leal et al. (2016).

Parameter	Value
Number of traders (N_L)	10,000
Trading frequency mean (θ)	20
Boundaries of trading frequency ($[\theta_{min}, \theta_{max}]$)	[10, 40]
Chartist's order size parameter (α_c)	0.04
Chartist's shock standard deviation (σ^c)	0.05
Fundamentalist's order size parameter (α_f)	0.04
Fundamentalist's shock standard deviation (σ^f)	0.01
Price tick standard deviation (σ^z)	0.01
Intensity of switching (ζ)	1
Resting order periods (γ^L)	20

5.1 Properties of the results

In the initialization, all agents have zero funds and zero assets. While the agents can individually gain and lose money, it is a zero-sum game, i.e. aggregated over all agents, the amount of funds and assets should stay at zero. In order to check this, I present a trajectory of the quantities for every time step. In Figure 7, the total amount of money per time step, normalized by the number of agents, is

Table 2: Parameter values of the HF trader as given in Jacob Leal et al. (2016).

Parameter	Value
Number of traders (N_H)	100
Activation threshold parameters ($[\eta_{min}, \eta_{max}]$)	[0, 0.2]
Resting order periods (γ^H)	1
Maximum order value	$\pm$3000
Maximum order size in percent of total volume in the opposite side of the LOB	25 %
Market volumes weight in trader's order size (λ)	0.625
Order price distribution support ($[\kappa_{min}, \kappa_{max}]$)	[0, 0.01]

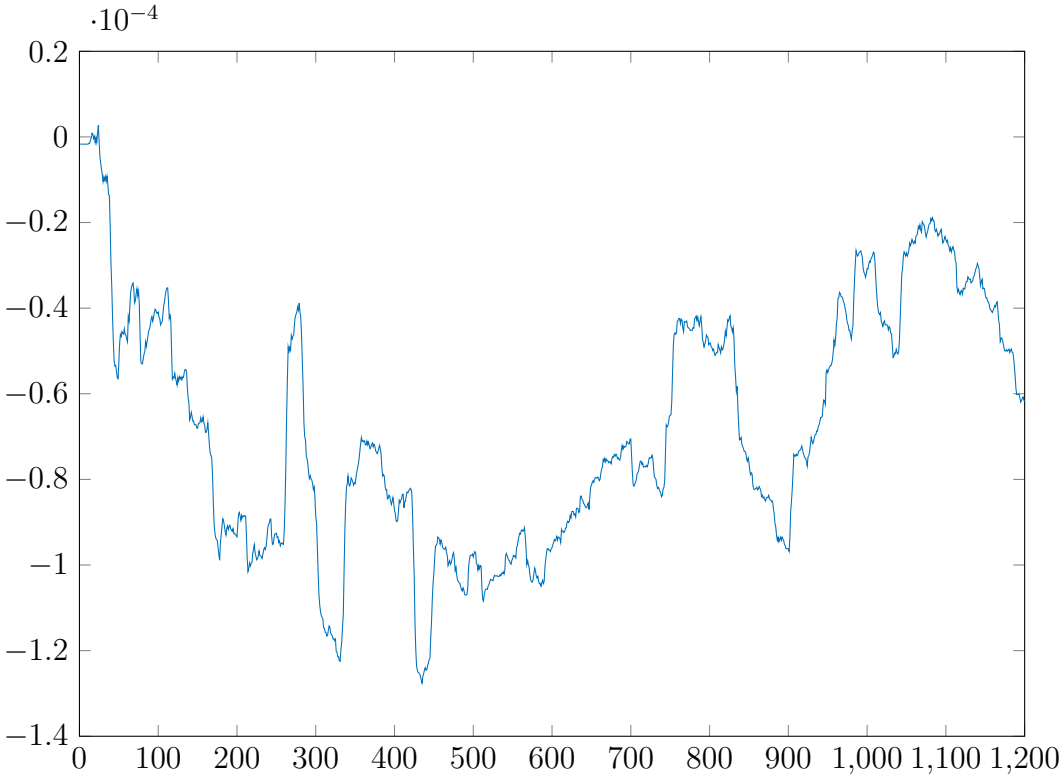
Table 3: Other default parameters provided by Jacob Leal et al. (2016).

Parameter	Value
Monte Carlo iterations	50
Number of trading sessions (T)	1,200
Fundamentalist value shock standard deviation (σ^y)	0.01
Fundamental value price drift parameter (δ)	0.0001

Table 4: Assumptions made in the replication simulation due to the lack of description in Jacob Leal et al. (2016).

Parameter	Value
Initial funds of all LF and HF traders	0
Initial assets of all LF and HF traders	0
Initial switching probability ($\Phi_{i,t=0}$)	0.5
Initial fundamental value	100
Asset prices of the last two time steps	[100, 100]

Figure 7: A trajectory of the total volume of the monetary funds per time step, normalized by the number of agents.

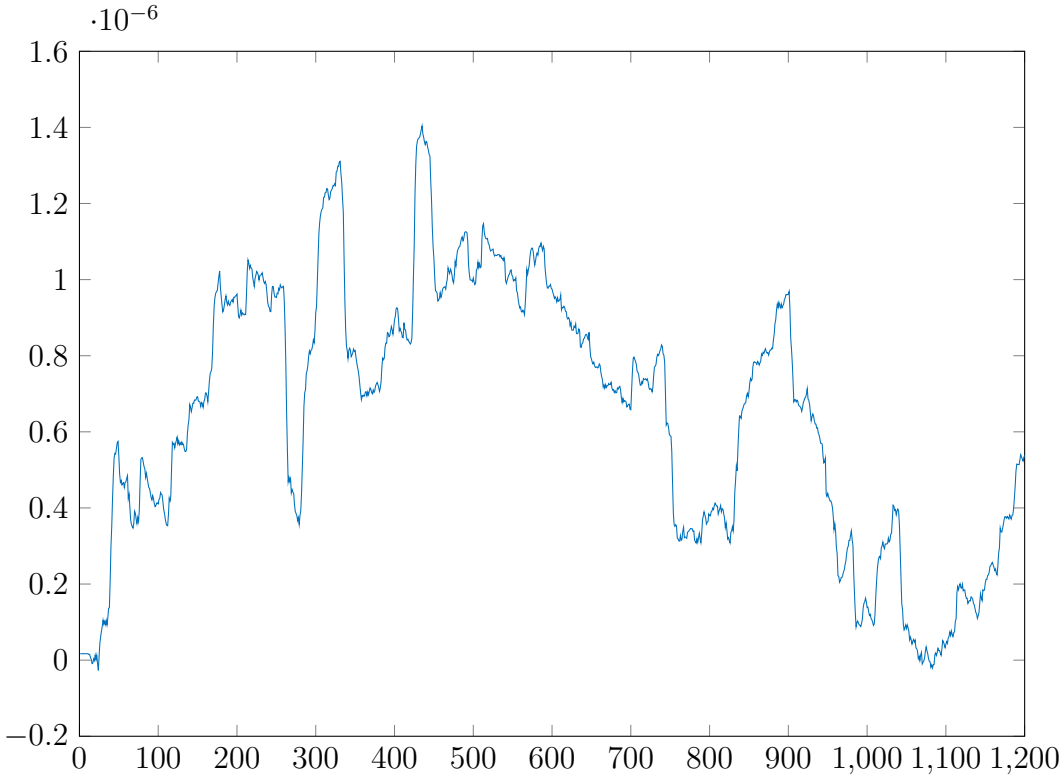


presented. The amount is in the order of magnitude 10^{-4}, which can be viewed as zero up to numerical inaccuracies.

The same can be said about the total amount of the asset which is present per time step (see Figure 8). The amount is in the order of magnitude 10^{-6}, which is again close enough to zero.

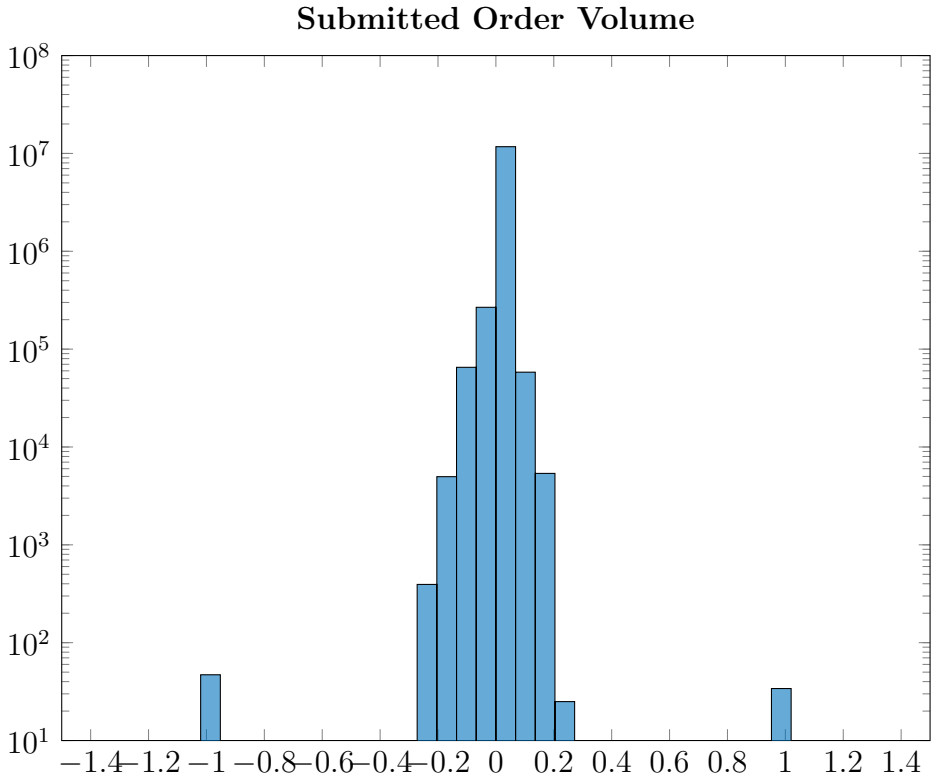
Figure 9 gives an insight on the submitted orders. It is a histogram of all submitted orders across all traders and all time steps. Note that this histogram has a log scale on the y-axis. An order volume of zero means that no order has been submitted at one particular time step by one particular agent. The prevalence of this occurrence adds up to 11.7 million times. Buy orders are represented as positive values, sell

Figure 8: A trajectory of the total volume of the asset per time step, normalized by the number of agents.



orders as negative values. One can observe that the frequency of buy and sell orders are somewhat balanced. Furthermore, as criticized earlier, the way the order volume calculation is defined implies that order volumes are typically less than 1, and mostly around or equal to zero, with the latter clearly dominating the histogram. It should be noted that at plus and minus one, a large amount of orders are submitted, which looks rather like an artifact of the truncated distributions. Unfortunately, neither Jacob Leal et al. (2016) nor Platt & Gebbie (2016) presented how the submitted orders in their simulations looked like. Therefore, no comparison of their results regarding the submitted orders to my replication of it can be made.

Figure 9: A histogram of the volume of the submitted orders, where zero means that no order has been submitted, positive values are buy orders and negative values are sell orders.



5.2 Replication of the Stylized Empirical Facts of Financial Markets

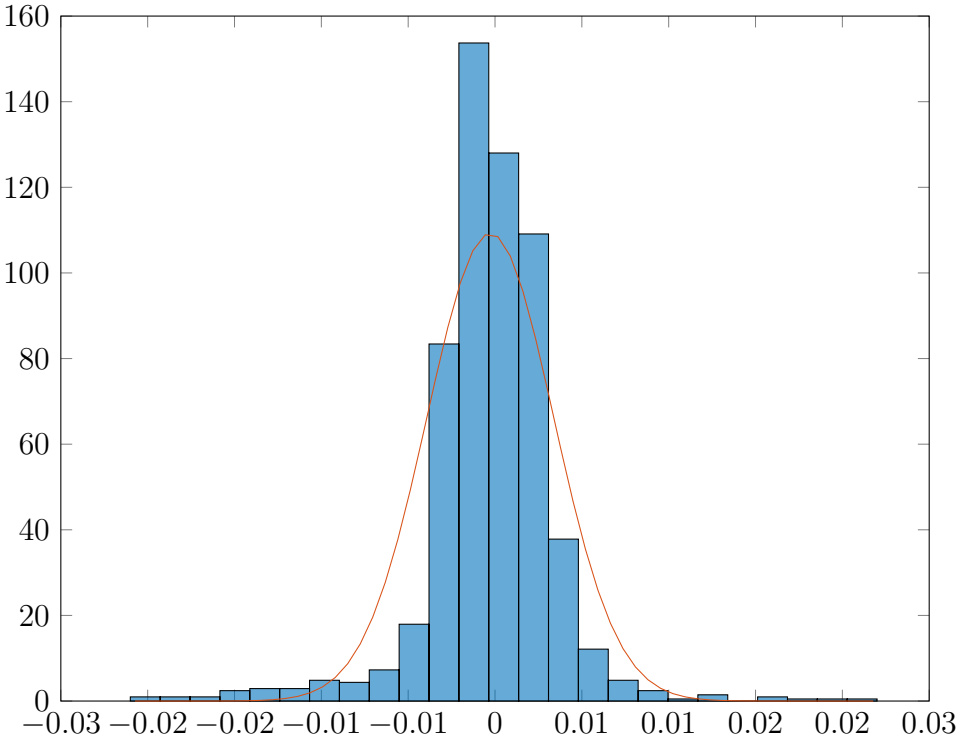
According to Cont (2001), the term "stylized empirical facts" describes the properties that asset prices, while varying across instruments, time phases, and markets, still share with each other. These stylized facts properties can be used to validate whether the model simulates an environment which is close enough to a real financial market.

5.2.1 Log Return Distribution Properties

According to Cont (2001), the distribution of returns is usually fat-tailed when compared to a normal distribution. The returns can be derived by taking the logarithmic difference of two prices. In Figure 10, the histogram of the model-generated log returns across all trading sessions shows clear fat tails. The same property is found in the model of Jacob Leal et al. (2016) (Figure 3, note that this is a logarithmic scale on the y-axis) and the replicated model in Figure 4 of Platt & Gebbie (2016). However, the latter displays a histogram which is even more leptokurtic.

Furthermore, I use a Q-Q plot to compare the data samples of the simulated log

Figure 10: A histogram of the log returns obtained from the simulated asset prices across all trading sessions, where the red line indicates the fitted normal density function.



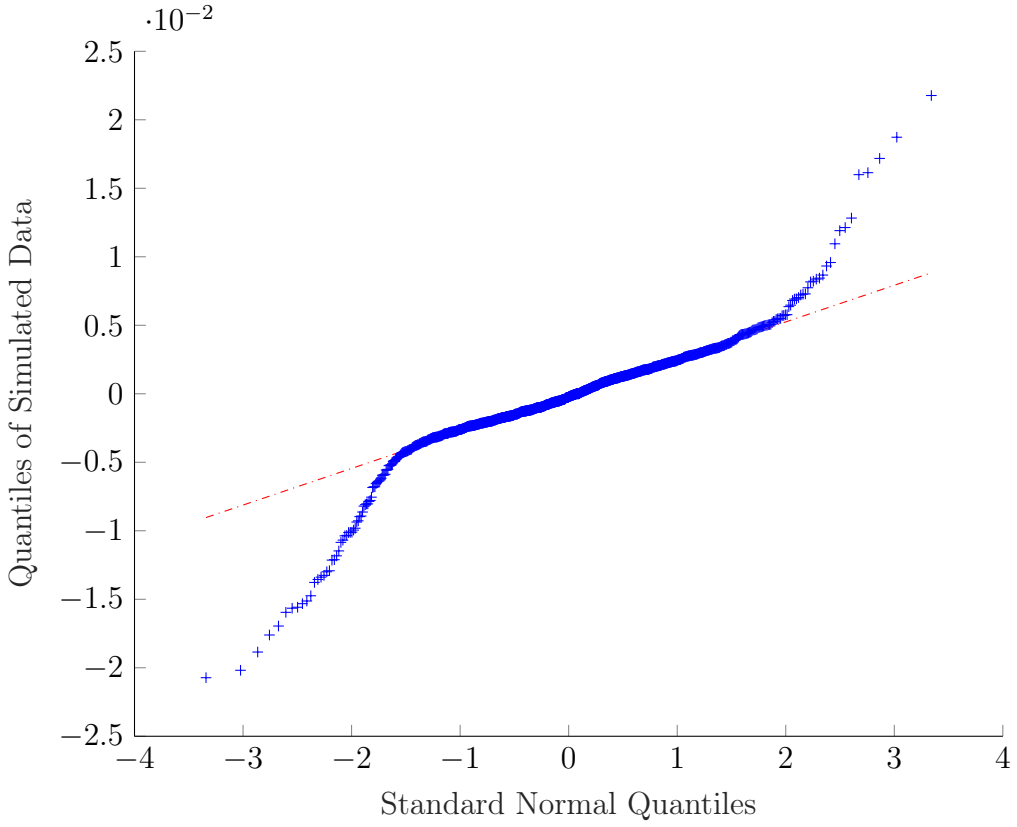
returns to a standard normal probability distribution. A Q-Q plot graphically shows whether two data sets stem from a population with the same distribution. In Figure 11, the quantiles of the standard normal distribution (on the x-axis) correspond to the red y=x line. Now, a blue point corresponds to one of the quantiles of the log returns distribution, plotted against the quantiles of the standard normal distribution. As the blue plots deviate from the y=x line, it is clear that the log returns are not perfectly normal distributed. In particular, the two deviating ends on the left and on the right side indicate that the log return distribution is heavily fat-tailed. Furthermore, the presence of the slightly larger tail towards the negative region indicates a stronger skewness towards the left. This suggests that extreme negative events are more probable than extreme positive events. This Q-Q plot resembles very much to the one in Platt & Gebbie (2016) (Figure 5).

5.2.2 Log Return Autocorrelation Properties

Another stylized fact listed by Cont (2001) is the absence of significant autocorrelation in asset price returns. The author gives an intuitive explanation for this observation: If autocorrelation existed, i.e. if price changes were correlated to each other, one could use this information to follow a simple investment strategy with positive expected earnings, which would be "statistical arbitrage" (Cont 2001, p. 229). These strategies cause correlations to diminish.

In my simulation, the autocorrelation values do not follow any significant pattern and do not differ significantly from zero, as can be seen in Figure 12. The red points display the magnitude of autocorrelation. While the autocorrelation is larger when the time lag is small, it rapidly decreases until the number of lags is 6. Thereafter, the magnitude of autocorrelation mostly stays inside the confidence bounds, depicted by the blue lines. The 95% confidence bounds in this simulation turn out to be $[0.0578, -0.0578]$. According to Cont (2001), the phenomenon of

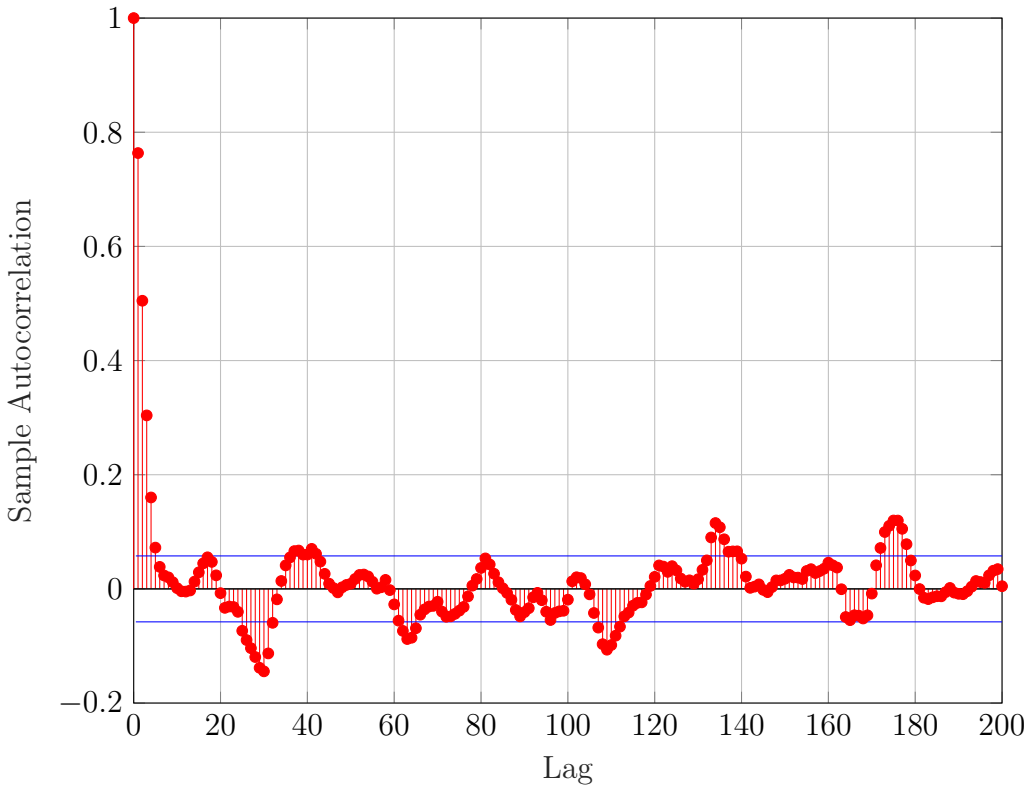
Figure 11: A Q-Q plot that displays the quantiles of the simulated log returns against the quantiles of a normal distribution.



autocorrelation being higher at short time lags can also be observed in actual markets. These short intraday time lags represent the time the market takes to react to new information.

In my replication, the autocorrelation of *absolute* log returns keeps a steady magnitude, instead of slightly decaying throughout the number of lags (Figure 13). This is not in line with Cont (2001)'s stylized facts. Cont (2001) observes that the autocorrelation function of absolute returns decays slowly as a function of the time lag, which can be viewed as an indicator for long-range dependence. Jacob Leal et al. (2016) does succeed in presenting decaying autocorrelation.

Figure 12: Log-return sample autocorrelation function.



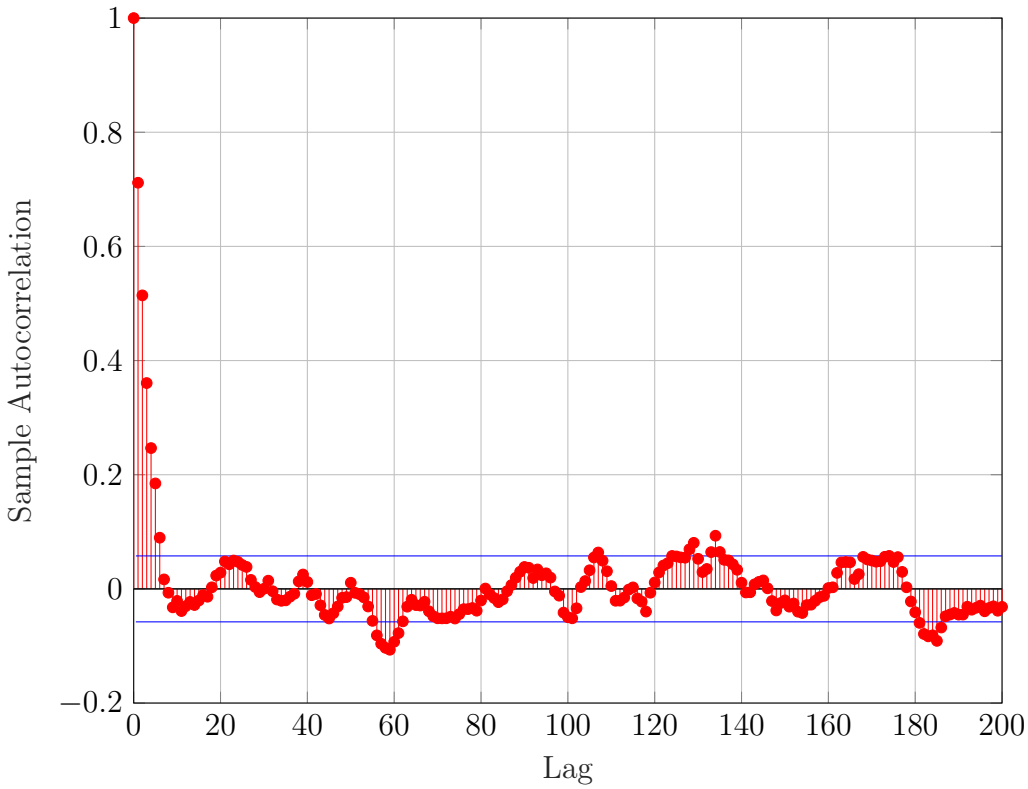
5.3 Alternative Modules

In this section I present the results from the simulations when using alternative implementation.

5.3.1 Alternative Closing Price Definitions

As announced earlier, I attempt to investigate the effect on the stylized facts of financial markets by modifying the way of determining closing prices. I implement five different methods: the last price as the closing price, which is the default version, and four alternatives, two of which are claimed to make no difference in Jacob Leal et al. (2016).

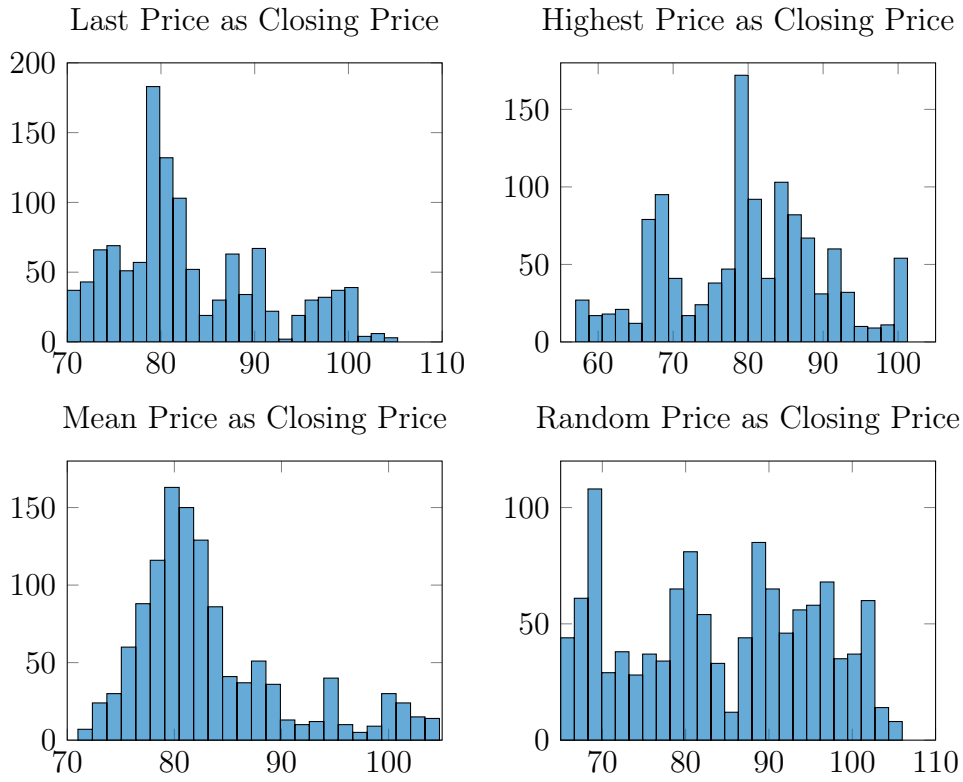
Figure 13: Sample autocorrelation function with absolute log returns.



Last Price as Closing Price This is the version implemented in Jacob Leal et al. (2016) and Platt & Gebbie (2016) and is the default method in my replication. Intuitively, this definition makes sense, as in real markets, the price of the very last transaction of a day is the closing price. As presented earlier, this method generates stylized empirical facts which are in line with empirical observations. Therefore, this method seems to be suitable for the simulation.

Highest Price as Closing Price In an earlier version of Jacob Leal et al. (2016) the closing price was defined as the maximum price of all executed trades of the current period, which seems quite unintuitive. In real markets, prices do not reliably increase towards the end of a time period, let alone reach its peak. This

Figure 14: Comparison of the asset price histograms.



method creates the asymmetry that profits are always non-negative, as profits are calculated by taking the difference between the closing price and the individual trading price. Per definition the closing price is larger or as large as the individual trading price.

Jacob Leal et al. (2016) claim that there is no difference in the simulation when using either method. However, in my implementation the results differ. A histogram of the closing prices can be seen in Figure 14. While the histograms of "Last Price as Closing Price" and "Highest Price as Closing Price" do look alike (with prices being slightly lower in the latter case), the distribution of the log returns differ significantly, as depicted in Figure 15.

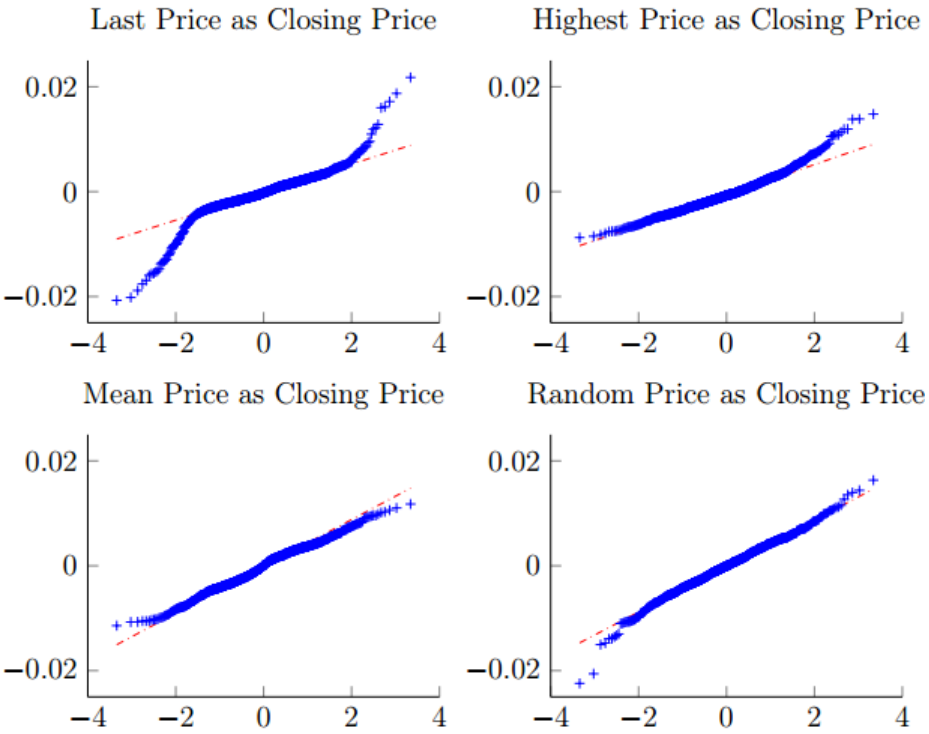
The distribution of the "Highest Price as Closing Price" is significantly more dis-

Figure 15: Comparison of the log return histograms.

persed and positive extreme events seem to be far more probable than negative ones. The Q-Q plots in Figure 16 confirm that the sample distribution is right-skewed.

Mean Price as Closing Price In Jacob Leal et al. (2016), the authors claim that not only the method of highest price as closing price, but also the average price as closing price does not change any results. When considering the log return distribution in Figure 15, however, one sees that it differs considerably from the default case distribution. The Q-Q plot in Figure 16 also indicates that the tails are thin, which is opposite of what Cont (2001) observes in real markets.

Figure 16: Q-Q plot comparison.



Lowest Price as Closing Price Although not mentioned in Jacob Leal et al. (2016), I implement the alternative "Lowest Price as Closing Price" to test the robustness of the simulation to a change in this direction. This method is the counterpart to "Highest Price as Closing Price". Here, out of all executed trades of the current period, the lowest price is used as the closing price. In this case the simulation completely breaks down. After a few hundred time steps in the simulation, every last order in the order book gets matched, and thus, the order book is empty. An empty order book, however, means that there exists no current best price. Without a specified best price, however, HF traders cannot set their limit prices (see Formula 3), and hence, their strategies are *undefined*. This causes the simulation to throw an error. Of course, the code could be implemented in such a way that it catches and handles this error. But one needs to keep in mind that at this point, no inference of the closing price can be made. If no one is neither

willing to set a buy nor a sell order in the order book, there literally exists no price for it. So if one would prefer to make the simulation to keep going, one would have to manually set an assumption of the closing price value. However, whatever assumption regarding this price is made, it is hard to justify why it would model the real world. The closing price is neither 0, nor infinite, nor negative infinite - it is simply undefined. Therefore, any further results of this simulation would be meaningless and therefore useless.

Random Price as Closing Price The method "Random Price as Closing Price" is not mentioned in Jacob Leal et al. (2016) either. Intuitively, this method is quite similar to the default version ("Last Price as Closing Price"), as neither extreme point is used (minimum or maximum) nor does it always use the mean. Instead, any price can be picked as the closing price. This logic is similar to the default method. Figures 15 and 16 confirm that the two methods are similar. The log returns distribution of this alternative resembles the default version log return distribution much more than any of the other alternatives. The Q-Q plot of "Random Price as Closing Price" shows that the log return distribution is fat tailed, with heavier tails in the negative area.

5.3.2 Alternative Volume Estimations for Profit Calculation

As announced earlier, I add an extension in which the way of profit calculation is changed. Instead of multiplying the difference of closing price and individual trading price with the offered quantity, it is multiplied with a weight between zero and one to account for the fraction of the submitted order actually being traded. Due to the lack of data on the true distribution of this fraction, I assume that this is a uniform distribution, although of course this assumption is not founded on any empirical observations. The uniform distribution merely serves as a placeholder

Figure 17: Comparison of the closing prices histograms.

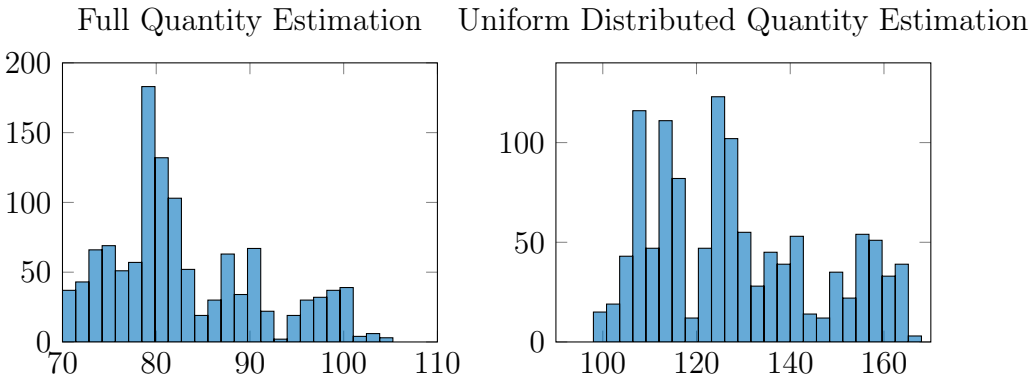
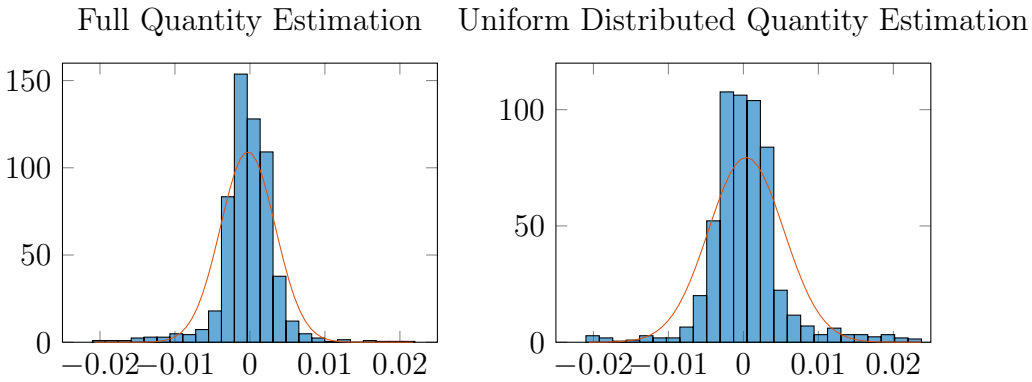


Figure 18: Comparison of the log return histograms.

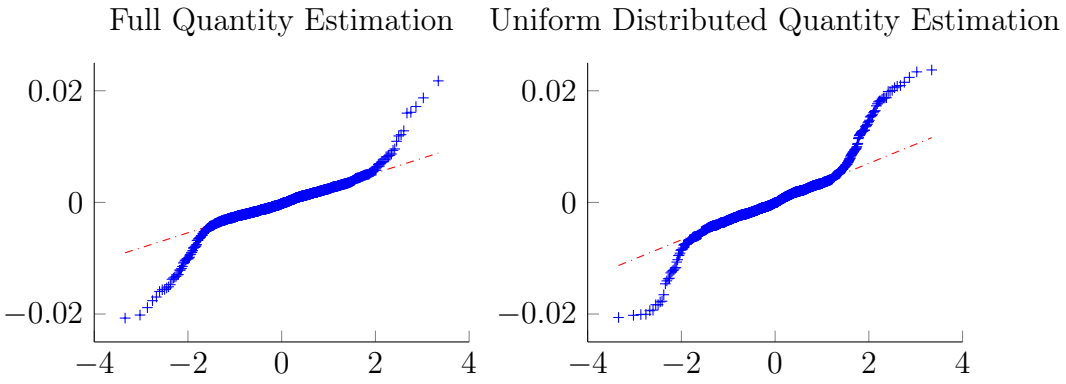


for the real underlying distribution and enables testing of the model's elasticity when changing the profit calculation formula.

The histogram of the closing prices in Figure 17 shows that the prices differ significantly, which however does not render the method invalid. It is more important to consider the log return distributions (see Figure 18) and the Q-Q plot comparison (see Figure 19). One can observe that under the alternative method, the log return distribution is leptokurtic and fat-tailed, which according to Cont (2001) exhibits the same stylistic facts as real financial markets.

Although the uniform estimation offers a more intuitive model for real world vol-

Figure 19: Comparison of the Q-Q plots.



ume estimations, no significant difference to the simple, full-quantity estimation can be seen. This might indicate a shortfall in the strategy selection process of the LF trader or simply indicate that the model isn't sensitive enough to pick up on such subtleties.

5.3.3 Continuous Matching

In an attempt to face the critics by e.g. Platt & Gebbie (2016) that "batch matching" is not realistic, I implement a switch in the model that allows the user to choose whether orders should be collected and then executed by the end of the time period or whether they should be matched and executed directly upon arrival. It turns out that in this modification as well, the market is degenerate and the model breaks apart. This can again be explained by the order book being empty. Evidently, when processing orders on a continuous basis, orders in the order book get matched before new ones can come in, resulting again in *undefined* best prices and in the simulation throwing an error. This might be avoided by drastically increasing the amount of traders and starting the simulation with a full order book.

6 Conclusion

In this Master thesis I replicated the agent-based model as described in "Rock around the Clock: An Agent-Based Model of Low- and High-Frequency Trading" by Jacob Leal et al. (2016). I implemented the replication in a way that it is robust and can easily be extended by diverse types of agents, price calculations, order mechanics, matching algorithms, and many others. One way to validate the model is to observe its statistical properties and compare them to stylized empirical facts of real financial markets. While some of my extensions did fulfill this criterion, others did not. In some cases, the model even broke apart completely. This is troubling, as my implemented alternatives only entailed slight changes compared to the default version. This demonstrates how fragile this agent-based model is. Minor changes in the implementation can greatly change the result, as happened here when defining the lowest price as the closing price or when having the orders be matched directly when arriving instead of batch-wise.

All in all, agent-based models seem to have helped researchers to better understand the outcome of the interplay of agents in various research fields. Nevertheless, one needs to keep in mind the limitations of agent-based models.

References

Brock, W. A. & Hommes, C. H. (1998), 'Heterogeneous beliefs and routes to chaos in a simple asset pricing model', *Journal of Economic dynamics and Control* **22**(8), 1235–1274.

Brogaard, J. (2010), 'High frequency trading and its impact on market quality', *Northwestern University Kellogg School of Management Working Paper* **66**.

Carrion, A. (2013), 'Very fast money: High-frequency trading on the nasdaq', *Journal of Financial Markets* **16**(4), 680–711.

CFTC-SEC (2010), 'Staff report, findings regarding the market events of may 6', https://www.sec.gov/news/studies/2010/marketevents-report.pdf.

Chen, S.-H., Chang, C.-L. & Du, Y.-R. (2012), 'Agent-based economic models and econometrics', *The Knowledge Engineering Review* **27**(2), 187–219.

Cohen, S. N. & Szpruch, L. (2012), 'A limit order book model for latency arbitrage', *Mathematics and Financial Economics* pp. 211–227.

Cont, R. (2001), 'Empirical properties of asset returns: stylized facts and statistical issues', *Quantitative Finance* **1**, 223–236.

Crockford, D. (2008), *JavaScript: The Good Parts*, first edn, O'Reilly Media, Sebastopol, CA, USA.

Daggett, M. E. (2013), *Expert JavaScript*, first edn, Apress Media, CA, USA.

Hagströmer, B. & Norden, L. (2013), 'The diversity of high-frequency traders', *Journal of Financial Markets* **16**(4), 741–770.

Hasbrouck, J. & Saar, G. (2013), 'Low-latency trading', *Journal of Financial Markets* **16**(4), 646–679.

Jacob Leal, S. & Napoletano, M. (2017), 'Market stability vs. market resilience: Regulatory policies experiments in an agent-based model with low-and high-frequency trading', *Journal of Economic Behavior & Organization* .

Jacob Leal, S., Napoletano, M., Roventini, A. & Fagiolo, G. (2016), 'Rock around the clock: An agent-based model of low-and high-frequency trading', *Journal of Evolutionary Economics* **26**(1), 49–76.

Jovanovic, B. & Menkveld, A. J. (2016), 'Middlemen in limit-order markets', *SSRN Working Paper* .

JSHint (2017), 'Jshint, a static code analysis tool for javascript', http://jshint.com/about/. Accessed: August 7, 2017.

Kirilenko, A., Kyle, A. S., Samadi, M. & Tuzun, T. (2017), 'The flash crash: High-frequency trading in an electronic market', *The Journal of Finance* .

Kobayashi, S. & Hashimoto, T. (2011), 'Benefits and limits of circuit breaker: Institutional design using artificial futures market', *Evolutionary and Institutional Economics Review* **7**(2), 355–372.

LeBaron, B. (2002), 'Building the santa fe artificial stock market', *Physica A* .

Mandelbrot, B. (1963), 'The variation of certain speculative prices', *The Journal of Business* **36**(4), 394–419.

Maslov, S. (2000), 'Simple model of a limit order-driven market', *Physica A: Statistical Mechanics and its Applications* **278**(3), 571–578.

Mizuta, T. (2016), 'A brief review of recent articial market simulation (multi-agent simulation) studies for financial market regulations and/or rules', *SSRN Working Paper* .

Network, M. D. (2017), 'Introduction to the dom', https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Accessed: August 7, 2017.

Nodejs (2017), 'About nodejs', https://nodejs.org/en/about/. Accessed: August 7, 2017.

Osmani, A. (2012), *JavaScript Design Patterns*, first edn, O'Reilly Media, Sebastopol, CA, USA.

Paddrik, M. E., Hayes, R., Todd, A., Yang, S. Y., Scherer, W. & Beling, P. (2011), 'An agent based model of the e-mini s&p 500 and the flash crash'.

Pellizzari, P. & Westerhoff, F. (2009), 'Some effects of transaction taxes under different microstructures', *Journal of Economic Behavior & Organization* **72**(3), 850–863.

Platt, D. & Gebbie, T. (2016), 'The problem of calibrating a simple agent-based model of high-frequency trading', *arXiv preprint arXiv:1606.01495* .

Preis, T., Golke, S., Paul, W. & Schneider, J. J. (2006), 'Multi-agent-based order book model of financial markets', *EPL (Europhysics Letters)* **75**(3), 510.

Steyer, R. (2014), *JavaScript: Die universelle Sprache zur Web-Programmierung*, first edn, Carl Hanser Verlag, Munich, Germany.

Veryzhenko, I., Arena, L., Harb, E. & Oriol, N. (2016), A reexamination of high frequency trading regulation effectiveness in an artificial market framework, *in* 'Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection', Springer, pp. 15–25.

Westerhoff, F. H. (2008), 'The use of agent-based financial market models to test the effectiveness of regulatory policies', *Jahrbücher für Nationalökonomie und Statistik* **228**(2-3), 195–227.

Xiong, Y., Yamada, T. & Terano, T. (2015), Comparison of different market making strategies for high frequency traders, *in* 'Winter Simulation Conference (WSC), 2015', IEEE, pp. 324–335.

Xue, Y., Gençay, R. & Wang, C. (2016), 'High frequency trading, noise herding and market quality', *SSRN Working Paper* .