University Name

Master's Thesis

Thesis Title

Author:
John Smith

Supervisor:
Dr. James Smith

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Research Group Name
Department or School Name

December 2013

Declaration of Authorship

I, John Smith, declare that this thesis titled, 'Thesis Title' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____________________________________

Date: ____________________________________

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

UNIVERSITY NAME (IN BLOCK CAPITALS)

Abstract

Faculty Name
Department or School Name

Doctor of Philosophy

Thesis Title

by John Smith

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor…

Contents

Declaration of Authorship i

Abstract iii

Acknowledgements iv

Contents v

List of Figures vi

List of Tables vii

Abbreviations viii

Physical Constants ix

Physical Constants x

Symbols xi

1 High Frequency Trading **2**
- 1.1 Overview of High Frequency Trading 2
  - 1.1.1 2

2 Model **3**
- 2.1 Model 3
  - 2.1.1 Asynchronous vs game theoretic model 3
  - 2.1.2 Overall architecture 3
  - 2.1.3 Modeling delays 4
- 2.2 Market components 4
  - 2.2.1 Stocks 4
    - 2.2.1.1 Fundamental price 5
  - 2.2.2 Messages 5
    - 2.2.2.1 Market information 6
    - 2.2.2.2 Orders 7
    - 2.2.2.3 Transaction receipts 7

2.2.2.4 Order cancellations 8
2.2.3 Order book 8
2.2.3.1 Price updating 8
2.2.3.2 Order matching 10
2.2.3.3 Empty order book 10
2.2.4 Market 10
2.2.4.1 Short selling 10
2.3 Agents 11
2.3.1 Slow traders 12
2.3.1.1 Arrival of orders 14
2.3.2 Market makers 14
2.3.3 HFT Chartists 15
2.4 Simulation rounds 15
2.5 Implementation 15
2.6 Overview of model parameters 15
2.7 Experiments 15

3 Parameter tuning **16**
3.1 Motivation and overall procedure 16
3.1.1 Selecting fixed parameters 18
3.2 Inverse simulation with a genetic algorithm 18
3.2.1 Representing parameters as genes 19
3.2.2 Model fitness 19
3.2.3 Genetic algorithm parameters 19
3.2.4 Controlling market behavior 20
3.2.5 Filtering parameters 20
3.2.6 Handling failed simulations 22
3.3 Applying the genetic algorithm 22
3.3.1 Time complexity 22
3.3.2 Experiments: Dividing the search into parts 23
3.3.3 Gene pool as data set 24
3.3.4 Convergence of parameters 25
3.4 Data analysis tools 25
3.4.1 Data visualization 25
3.4.1.1 Color toned scatter plots 25
3.4.2 Preprocessing 25
3.4.2.1 Handling outliers 25
3.4.2.2 title 27
3.4.3 Clustering algorithms 27

4 Experiments **28**
4.1 D3 28
4.2 D9 28
4.2.1 Calculating statistics for outliers 28

A Additional tables **30**

A.1 Dataset 1 30

B Third party software **31**

Bibliography **32**

List of Figures

2.1 **:** 5

2.2 6

2.3 When an agent wants to submit an order it has to go through several steps of interaction with the market. The process is comparable to how real traders communicate with markets via a network, such as the Internet. 7

2.4 9

2.5 The HFT agent submits a sell order for 100 stocks, and another agent submits a price-matching buy order which fills the sell order. Before the transaction receipt reaches the seller, the seller decides to cancel the order, and submit another order at a different price. When the transaction receipt reaches the seller, the agent promptly sends out a cancellation of its second sell order, as it knows it cannot fulfill the order. However, before the cancellation reaches the market, a third agent fills the sell order, and a receipt is send to the seller who ends up being short. 11

2.6 Slow trader order arrival 14

3.1 **Motivatoin for tuning:** the two 17

3.2 Example of a simulation which is assigned fairly good fitness values, but which was executed with clearly unrealistic parameters: $\mathtt{n}_{\mathtt{mm}} = \mathtt{n}_{\mathtt{sc}} = 0$. The simulation reaches the new fundamental price fairly quickly without any undershoot, and stays within the stability margin. The only point where it scores badly is the standard deviation which is slightly high due to the fluctuating trade price. 21

3.3 Color toned scatter plots of f_{s}, $f_\sigma$and f_{m}taken from dataset d1after applying log-scaling and manually removing outliers 27

List of Tables

2.1 TABLE ILLUSTRATING ORDER BOOK 8

3.1 Overview of parameters used in the genetic algorithm 20
3.2 Overview of experiments 20
3.3 An example data matrix containing the parameters of ten individuals who lived sometime during the execution of the genetic algorithm. In this case, each individual contained parameters for the number of HFT agents, as well as the latency and thinking time parameters. Hence, the data matrix has a column for each parameter. 24
3.4 This table contains the fitness values for each individual in table 3.3. Note that, in order to increase the reliability of the fitness measure of an individual, the recorded fitness-values are the average of the fitness-values obtained by evaluating each individual ten times 25

4.1 CAPTION 29

Abbreviations

HFT	**H**igh **F**requency **T**rader
ST	**S**low **T**rader
OB	**O**rder **B**ook
MM	**M**arket **M**aker

Physical Constants

Term	Explanation
ask price	
bid price	
fitness	
fundamental	
limit order	
market order	
liquidity	
match	When a sell order and a bu order happen to have the same listed price, they are sai
order book	
partial match	Two orders which match, but have different volumes.
share	A fraction of ownership of an asset, such as a stock
spread	
standing order	A market order registered at an order book and waiting for a matching order
tick	
volatility	

Physical Constants

p^m	Time delay in rounds from agent i to market j. Note that $\tau_{i,j} = \tau_{j,i}$.
s_r	Spread at the end of round t
p_r^a	The lowest askprice in the order book at the end of round r
p_r^b	The highest bidprice in the order book at the end of round r
p^m	Match price, i.e., the price at which a trade is executed
f_r	Fundamental price at round r

Symbols

Symbol	**Description**
$\mathtt{n_{rounds}}$	Number of simulation rounds
***λ	Average number of ST orders per round
$\lambda_{c,\mu}$	Mean of the Gaussian distribution for HFT market chartist latency
$\lambda_{c,\sigma}$	Standard deviation of the Gaussian distribution for HFT market chartist
N_c	Number of HFT market chartist agents
$T_{c,\mu}$	Mean of the Gaussian distribution for HFT market chartist thinking time
$T_{c,\sigma}$	Standard deviation of the Gaussian distribution for HFT market chartist thinking time
$H_{c,\mu}$	Mean of the Gaussian distribution for HFT market chartist time horizon
$H_{c,\sigma}$	Standard deviation of the Gaussian distribution for HFT market chartist time horizon
$W_{c,\mu}$	Mean of the Gaussian distribution for the number of rounds that HFT market chartist ag
$W_{c,\sigma}$	Standard deviation of the Gaussian distribution for the number of rounds that HFT mark
$\lambda_{m,\mu}$	Mean of the Gaussian distribution for HFT market maker latency
$\lambda_{m,\sigma}$	Standard deviation of the Gaussian distribution for HFT market maker latency
N_m	Number of HFT market maker agents
$T_{m,\mu}$	Mean of the Gaussian distribution for HFT market chartist thinking time
$T_{m,\sigma}$	Standard deviation of the Gaussian distribution for HFT market chartist thinking time.
$\mathtt{n_{mm}}$	Number of hft market maker agents.
$\mathtt{n_{sc}}$	Number of hft simple chartist agents.
	Mean and standard deviation of MM spread
	Mean and standard deviation of MM order volume
	Mean and standard deviation of MM order length
	Mean and standard deviation of chartist window length
	Mean and standard deviation of chartist sensitivity
	Mean and standard deviation of chartist aggressiveness

Mean and standard deviation of chartist order volume

Mean and standard deviation of chartist trade frequency

For/Dedicated to/To my…

FOREWORD As such, this project turned out to be just as much about software engineering as Many models of artificial markets resemble typical game theoretic models in that all trading is round based. Typically all agents receive the market information and evaluate their strategies once every round. Such models are very useful when all agents are more or less equally fast, since it is then reasonable to assume that they have access to the same information.

The model proposed in this work is different from such models in that it assumes that agents generally do *not* have access to the same information when they evaluate their strategies.

Chapter 1

High Frequency Trading

1.1 Overview of High Frequency Trading

1.1.1

In the literature,

Chapter 2

Model

2.1 Model

As explained in section 1, perhaps the most distinguishable aspect of high frequency trading is the speed with which agents can react to new market information. It is therefore essential that a model should capture this aspect, if it is to be used to draw generalized conclusions about the influence of high frequency trading in the markets.

What are the goals of this model

What are not the goals of this model.

2.1.1 Asynchronous vs game theoretic model

In this work, we have tried emulate the asynchronous nature of a continuous auction by

As such, our model bears little resemblance to models derived from a game-theoretic basis, which typically

This work resembles a real-time simulation.

2.1.2 Overall architecture

The model consists a market and agents. Agents and the market communicate by exchanging messages which all arrive one or several rounds after they were issued. A complete simulation consists of the several consecutive rounds. In each rounds, some agents submit orders, while others wait for new market information. Order messages arrive at the order books, and trades are executed when prices match. The following sections will describe the model in detail.

2.1.3 Modeling delays

Although each round corresponds to a period of real-time, it is not particularly important to specify how long that period is. Instead, what matters is that there is a difference in speed between the agents. In other words, the important thing is that some agents are much faster than other agents. If one thinks of each round as a millisecond of real-time, one realizes that an agent simulating a human trader will require several thousands of simulation rounds to react to market news. On the other hand, fast algorithmic traders may only require a few rounds, making them several orders of magnitude faster than the slow traders.

This focus on the extremely

Another issue when simulating

2.2 Market components

Sending/receiving orders, supply liquidity

2.2.1 Stocks

A stock is an asset which is traded on a market. The price of a stock is a mysterious thing. A stock is only worth as much as people are willing to pay for it, and the price at which is it traded thus goes up and down according to what beliefs people hold. In financial markets, every trader is supposed to have access to the same information. However, two traders might disagree on the meaning of some piece of information. The way in which a trader evaluates market information and reaches a conclusion on how to trade is called a strategy. While any function which takes some information relevant to the market as input and gives a decision of how to trade (or not trade at all) can be termed a strategy, it is useful to divide strategies into two broad categories. In the first category are strategies which are dubbed chartist strategies, which basically tries to extrapolate on the past price movements. In the other category are strategies which

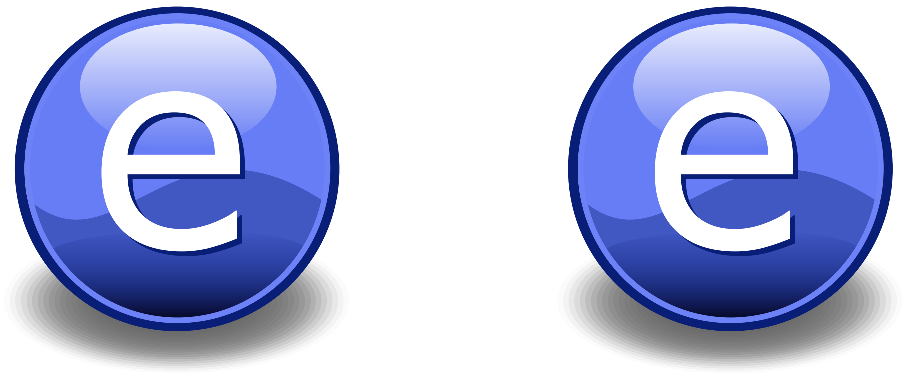

(A) (B)

Figure 2.1: :

are based on some analysis of the true value of the stock, called the fundamental value. Not surprisingly, these are called fundamentalist strategies.

Whether or not one type of strategy is more accurate than the other, it is a fact that both types are employed by traders. Hence, a model of such an environment needs to simulate both a traded price and a fundamental price.

2.2.1.1 Fundamental price

A common way to model the development of the fundamental price is to use a stochastic random walk process. The idea is that, assuming that markets are efficient, any available information about the stock is already reflected in the fundamental price. When some news arrive, this will quickly cause the price to change according to the nature of the news, as rational traders act to adapt to the new situation. The justification for modeling this with a random walk is that, since the fundamental price already reflect whatever news is available, it will only change as new information is released. In other words, the fundamental price is independent of past information. Since new information is fundamentally unpredictable, a random walk model seems suitable.

The idea behind this is that

2.2.2 Messages

All communication between the market and agents is transmitted in messages. A message sent from agent i to market j (or the other way around) has a transmission time of $\tau_{i,j} = \tau_{j,i}$. The smallest possible transmission time is $\tau_{i,j} = 0$. This means that no information is transmitted instantly between agent and market. $\tau_{i,j}$ is constant through the simulation. Several message types were implemented in order to accommodate the various types of communication.

Figure 2.2

2.2.2.1 Market information

One of the key points of simulating delays is that agents always trade on old information. Before an agent can evaluate its strategy, it has to request the most recent market information. In a model without delays, an agent would simply receive the state of the market in the current round, but when information is delayed the process is somewhat more cumbersome. First the agent sends off a request to obtain the information about the market state. When the request arrives at the market some rounds later, the market serves the request and by sending back another message containing the information. The contents of this message depends on the agent strategy, as the various agent strategies require different information. In the case of a single market, it is reasonable to simplify the model such that the market serves the request instantaneously, since any delay inherent in the market is common for all agents. After a further delay, the message containing the market state finally arrives at the agent, and the agent can then start evaluating its strategy. Figure 2.3 summarizes the procedure.

This is analogous to how

FIGURE 2.3: When an agent wants to submit an order it has to go through several steps of interaction with the market. The process is comparable to how real traders communicate with markets via a network, such as the Internet.

2.2.2.2 Orders

An order is a message which is sent from an agent to a market when the agent has decided to trade. An order is an offer to buy or sell a specified number of shares at a certain price at a certain market. Orders can either be limit orders or market orders. A limit order will only result in a trade to be executed if there is a matching order when it arrives to the market. A market order will stay in the order book until a matching order arrives, or until it expires after a number of rounds set by the submitting agent.

When an agents submits an order, it has to decide on the trade price, the number of shares, limit or market order, and whether to buy or sell. Details on how each type of agent does this can be found in section 2.3.

2.2.2.3 Transaction receipts

When two orders match, a receipt is sent to each of the two agents involved in the trade. The seller receives a receipt specifying the number of shares that it has to deliver, and the buyer gets a receipt for the amount of cash to be paid. Because of the transmission

ASK-volume	Price	BID-volume

Table 2.1: TABLE ILLUSTRATING ORDER BOOK

delay, the agents to not update their portfolios when the trade actually happens, but when they receive the receipt. In the case that an agent does not have enough shares or cash in its portfolio, the agent is allowed to borrow the necessary assets, thus bringing its portfolio into negative. An agent cannot submit new sell orders while holding a negative number of shares. Similarly, an agent cannot submit any new buy orders while having a negative amount of cash. In the case that the agent has neither cash nor shares, it simply becomes inactive.

2.2.2.4 Order cancellations

It can happen that an agent wants to change a previously submitted order, or cancel it entirely. In fact, this is what the market maker agent does frequently, as described in section 2.3.2. In this case, the agent issues a message to the market requesting that the order should be removed. Due to the presence of delays, the agent's order might be filled before the cancellation reaches the market, in which case the market will ignore the request to cancel.

2.2.3 Order book

The order book is a record of all unmatched orders for a single stock. Since any buy- and sell orders submitted at the same price will cause a trade to be executed, and the matched orders to subsequently be removed, there must at point of time during the simulationbe a non-negative price difference between the sell order with he lowest price and the buy order with the highest price. This difference is called the *spread*, and is denoted as follows

$$s_r = p_r^a - p_r^b \tag{2.1}$$

where p_r^a is the lowest ask price and p_r^b is the highest bid price, both at round r. These prices are also frequently referred to as the *best* ask and bid prices.

When an agent

2.2.3.1 Price updating

Each time an order is added or removed, the order book has to update the best bid and ask prices. Since it often happens that several orders arrive in the same round. This means that the order book spread can fluctuate within a single round. However, since one round is considered the minimum quantum of time, these within-round fluctuations are not recorded in the order book history. Instead, after all orders have been processed, the resulting best bid and ask prices are registered as the best prices for that round. Agents who look at the market will therefore only be able to see the state of the order book after the book has finished processing all price changes due to the arrival or removal of orders. The subscript denoting time equation 2.1 therefore refers to the prices at the end of that round. This difference between the traded prices and the best prices is shown on figure 2.4

Figure 2.4

When no orders arrive or are removed from the order book, the prices are updated as $p^a_{r+1} = p^a_r$ and $p^b_{r+1} = p^b_r$.

Since orders can be removed due to cancellations or because they expire, the order in which incoming messages is processed matters to the outcome of p^a_r and p^b_r. Messages are therefore processed in random orders, so that no agent is favored.

2.2.3.2 Order matching

When a trade is executed between orders o_1 and o_2, the traded volume is

$$\Delta v = \min(v_{o_1}, v_{o_1})$$

If $v_{o_1} = v_{o_2}$ the orders are *fully matched*, and both are removed from the order book. In the case of a *partial match*, that is, if the volume of one order is larger than the volume of the other, then the order with the smaller volume is removed, and the volume of the other order is subtracted by Δv. The price of the transaction is the price of the market order which was already in the book.

Each agent knows the volume of every order that it submitted, when the order was dispatched. However, when an order is partially filled by a matching but smaller order, the volume of the order at the market changes. Since it takes time for the order to be transmitted for the agent to the market, the agent cannot immediately update its knowledge of the order volume. In this case the order has one volume at the agent side and another in the market side. The momentary disparity of agent market side and agent side volumes can have several consequences, such as agents short-selling without, agents submitting cancellations for orders which have already been filled. Rules that handle these situations are described in section **??**. Unlike volumes, the price of a standing market order does not change, and hence the situation of a disparity between market- and agent-side price knowledge does not occur.

2.2.3.3 Empty order book

Since orders are removed when their volume is depleted, it can happen that one or both sides of the order book is empty. XXXWRITE SOME MORE HERE

2.2.4 Market

2.2.4.1 Short selling

Although some market do allow deliberate short selling, this practice is not allowed in the simulation. That is, an agent is not allowed to place a sell order for more stock than it has in its portfolio at the time it places the order. However, due to the presence of delays, it can happen that an agent is required to deliver on a sell order for more stocks than it holds when notified of the order. A sequence of events which causes this to happen is illustrated on figure 2.5. The agent who is short is required to deliver the stocks, and thus goes into negative on its portfolio, and has to buy back the stocks before it can place further sell orders. Although the sequence of events shown in figure 2.5 may seem unlikely, it did in fact occur frequently, making it necessary to implement handling of this special case.

FIGURE 2.5: The HFT agent submits a sell order for 100 stocks, and another agent submits a price-matching buy order which fills the sell order. Before the transaction receipt reaches the seller, the seller decides to cancel the order, and submit another order at a different price. When the transaction receipt reaches the seller, the agent promptly sends out a cancellation of its second sell order, as it knows it cannot fulfill the order. However, before the cancellation reaches the market, a third agent fills the sell order, and a receipt is send to the seller who ends up being short.

2.3 Agents

The model contains three types of agents, each employing a different strategy. Each strategy has several parameters which greatly impact the behavior of the agent.

The purpose of this work is to model a market in which some agents are much faster than other agents. To this effect, we divide the agents into two groups. The first one is the group of slow traders, which are meant to represent human traders, and algorithmic traders using long-term strategies. The other group is the strategies representing the high frequency traders.

[1]

2.3.1 Slow traders

The purpose of this agent type is to include agents which

The stylized trader model used in this work is inspired by the model used in [1] and [6]. However, due to the fundamental differences in the way that the simulation works, the model has been modified significantly.

Since the stylized trader model is so predominant in the market simulation literature (see [**?**], [**?**] etc.), we feel that we need to justify our choice of not using it.

The basic idea of the model is that there are basically three basic techniques that any trader mixes draws upon to form his own strategy.

Fundamental analysis Traders subscribing to this way of thinking believe that they can know the true value of a stock by estimating the fundamental price (see section 2.2.1.1). Furthermore, such traders believe that any deviation from the fundamental price is due to other traders misinterpreting the market, and that such deviations will eventually disappear. In other words, given enough time, the traded price will converge towards the fundamental price.

Technical analysis Traders using technical analysis do not care whether or not the stock is over valued. Instead, they believe that they can predict future price movements from past data. Traditional technical analyst approaches extrapolates on price movements by using simple mathematical models and a good deal of heuristics.

Noise trading Some traders are in possession of insider knowledge, which means that they think that they know something about the stock that others do not. They use this information to trade the stock, for better or for worse. Such traders were dubbed noise traders, since it is highly unlikely that any one individual would come in possession of information which actually gives that person an advantage in the market. Any such belief is therefore naive, and might as well inflict a loss on the agent than generate a profit.

Whether or not one strategy is better than the others is not for this thesis to discuss. The point is that all three techniques are commonly used among traders, and that rather to subscribing solely to one way of thinking, most people use a mix of all three.

In this model, it is assumed that all slow traders know the true fundamental value of the stock at some time in the past.

aspects of human behavior which influences their decisions to trade. The first one

with the one significant alteration that it uses no historical data. In other words, the model does not simulate the contribution made by chartist speculation. This is justified because of the short time-scales at which the HFTs operate.

As mentioned, chartist strategies work by extrapolating on historical price movements to predict future price movements. However, due to the short span of real-time which the simulation covers, it hardly makes sense to

In our model we preserve the fundamental and noise trader characteristics, but dispose of the chartist element. Although this is contrary to common practice, we justify it as follows. Most chartist strategies operate on a timescale of days to weeks to months. However, the simulation proposed in this work merely simulates a few minutes of real-time[1]. Any chartist strategy based on the accumulated history of price movement over a long period of time will simply be too sluggish to follow with the high frequency price fluctuations occurring within the simulation.

It is possible that some traders utilize chartist algorithms on a very short time scale, but any trader employing such a strategy must be fast enough to react to the rapid changes. In order for the model to cope with the presence of such agents, a HFT-chartist agents was implemented as described in section 2.3.3.

Secondly, we are not interested in long term market dynamics. The model is limited to simulating only periods which are of key interest, as is discussed in section 2.7.

However, remembering that the time resolution at which slow traders are observing the markets is so low, that the information that they are watching is close to constant during the simulation. The contribution from a chartist strategy can therefore be represented by a random number. The strategy is therefore simply

The stylized traders play the same role as the slow trades in [3]

The orders submitted the stylized traders throughout the run of the simulation should be thought of as being submitted by a variety of different agents, all observing the same date, but interpreting it differently using different strategies. As for the timing of the order, the same argument goes. In the simulation, a constant number of orders arrive at

[1]This is not entirely accurate as each round can be interpreted as an arbitrary length of real-time. However, since we are interested in what phenomena occur when we have some agents which are several orders of magnitude faster than other agents, we need a high time resolution, effectively capping the length of real-time which can be simulated

each round, and sent to the buy- and sell side with equal probability. Thus no attempt it made to model any kind of herding phenomenon, since there is no time in which such a thing could occur.

2.3.1.1 Arrival of orders

Figure 2.6: Slow trader order arrive randomly according to a Poisson process. The bars show the volume generated by slow traders. XXXWRITEMORE

2.3.2 Market makers

For the sake of simplicity, each market maker is only allowed to have one order at each side of the order book at the same time. The agent can therefore not stack orders on either side of the order book.

2.3.3 HFT Chartists

2.4 Simulation rounds

2.5 Implementation

2.6 Overview of model parameters

2.7 Experiments

WRITE ABOUT KEY POINTS OF INTEREST IN TIME

Chapter 3

Parameter tuning

The model has several parameters which must be selected carefully before the simulation can be used to infer knowledge about market behavior.

The parameter tuning turned out to consume a significant amount of time, and simple using a genetic algorithm to optimize over the entire space of parameters was not enough. Instead, the process was a slow and iterative one of running the genetic algorithm to create a data set, analyze the data set to find out what was discovered in the search, and then run the genetic algorithm again with different parameters. Thus several data sets were created, each with the purpose of examining some aspect of the simulation, or of the parameter tuning method itself.

This chapter will cover the instruments used in the optimization of the model parameters, and also mention the machine learning tools used in the analysis of the data sets.

The parameter tuning has two overall goals, which are covered in the following section.

3.1 Motivation and overall procedure

First of all, the model must be calibrated such that it mimics the behavior of real markets. Since virtually every aspect of the simulation behavior depends on the values on the various parameters, these must be chosen carefully in order for the simulation to produce realistic behavior. An example of a simulation untuned parameters causing unrealistic behavior is given in figure 3.1a. Selecting realistic parameters is by far a simple task. First of all, it requires a way of quantifying the quality of each simulation. The choice of such a quantification is discussed in section 3.2.2. Secondly, there might be several different parameter configurations which produce seemingly realistic behavior,

(A) Parameters causing unrealistic dynamics

(B) Unrealistic parameters causing realistic dynamics

FIGURE 3.1: **Motivatoin for tuning:** the two

but do not correspond to a realistic market setting. An example for this is given in figure 3.1b, and section 3.2.5 briefly discusses this point.

The second goal of the parameter tuning is to find parameters which promotes certain desirable behaviors. For instance, we might be interested in determining which parameters causes the traded price to stabilize faster after a shock to the fundamental price. Metrics for doing this is discussed in section 3.2.2

The selection of parameters is a fairly complicated process because of the large parameter space, and because it takes a significant time to evaluate the fitness of a given set of parameters[1]. amount of time to execute a simulation. Because of this, the following three-step parameter selection procedure was used.

1. Fix some of the model parameters in order to reduce the search space for the optimization algorithm. This requires us to consider which parameters can be fixed without losing opportunity to gain insight into market behavior. Essentially this step is a question of prioritizing the optimization of some parameters over the optimization of others.
2. Use an optimization algorithm to find sets of parameters which yield realistic model behavior. A genetic algorithm was chosen for this purpose, and the details are explained in section 3.2.
3. From the set of parameter combinations found by the optimization algorithm, remove the parameter combinations which obviously do not correspond to a realistic setting.

[1]The calculation time depends largely on the parameters, such as the number of agents and how active these are. Typically one to several minutes are required to evaluate a single set of parameters.

3.1.1 Selecting fixed parameters

The main parameters of interest are the ones that control the latency and speed of the agents. The agent strategy parameters are less important, since

$\texttt{n}_{\texttt{rounds}}$ Due to the computational cost of running the simulation for a large number of rounds, the the number of rounds is fixed at 10^5 for all experiments.

Order volumes As with most of the other agent parameters, the

The remaining model parameters will either be fixed for each experiment, or varied by the genetic algorithm.

3.2 Inverse simulation with a genetic algorithm

Inverse simulation refers to the technique of specifying metrics measuring model behavior, and then using an optimization algorithm to search for parameters resulting in desirable (or undesirable) behavior.

In this work, a genetic algorithm was used to search the parameter space. The algorithm proceeds as explained below.

1. Generate a population of healthy individuals, e.g., individuals with valid parameters.
2. Evaluate fitness for every individual in the population.
3. Repeat n_{gen} times
   (a) Generate offspring by crossing existing individuals.
   (b) Apply mutation to with a certain probability to each individual (parents as well as children)
   (c) Evaluate fitness of children and mutated parents.

Mutation and crossover are the operators responsible for generating variation in the population, while the selection is responsible for propagating promising individuals to future generation where they might be improved. Several possible methods of performing each of these three steps exist in the literature (see[**?**], [**?**]), and section 3.2.3 briefly covers the method and parameters of the genetic algorithm.

3.2.1 Representing parameters as genes

Since the choice of mutation and crossover operators depends on the nature of the genes, the first step towards utilizing to search the model parameters is to decide on how to encode the parameters as individuals. A set of parameters is represented by an individual, i, consisting gene for each parameter, represented by a floating point $g_{i,j}$, where j denotes the index of the parameter. When the population is initialized, each $g_{i,j}$ is drawn from a uniform distributed in the range $g_j \in [0;1]$:

$$g_{i,j} \sim \mathcal{U}(0,1) \tag{3.1}$$

Some of the model parameters are integers, such as $\mathtt{n_{mm}}$and $\mathtt{n_{sc}}$, and these are rounded after being scaled and before they are passed to the simulation.

3.2.2 Model fitness

In order to use inverse simulation, it is necessary to decide on how to measure the quality of an instance of the simulation. In this work, the overall goal is to examine which parameter values cause the market to be stable, and which cause it to be unstable.

Another interesting point is the speed with which the market responds to the shock to the fundamental price, and which parameters influence this property. Furthermore, we are interested in investigating whether or not

- 'fit
- Are there certain parameter combinations which cause the market to behave in certain ways. S

In particular the parameters controlling various time delays are of interest. The search space of the parameters is very large, which makes an exhaustive search impossible. To this end, four fitness measures were defined. The balance between the number Several parameters influence the number of orders submitted by the high frequency traders.

3.2.3 Genetic algorithm parameters

Although this is a basic version of genetic algorithm, using it correctly is not necessarily easy, as was encountered. First of all, the parameters for the genetic algorithm itself must be established. The larger and more complex the search space, the more resources

Parameter	Assignment
Number of generations	200 to 1000
Number of individuals	100 to 1000
Cross-over points	2
Tournament size	3
Mutation probability	0.1
Mutation distribution	$\mathcal{N}(\mu = 0, \sigma = 0.1)$

TABLE 3.1: Overview of parameters used in the genetic algorithm

Parameter	**Range**
$\mathtt{n_{mm}}$	
$\mathtt{n_{sc}}$	

TABLE 3.2: Overview of experiments

the search will require, since the evaluating the fitness function (i.e., the running the simulation) will have to be done a larger number of times.

Table 3.1 presents an overview of the parameters used in the genetic algorithm.

3.2.4 Controlling market behavior

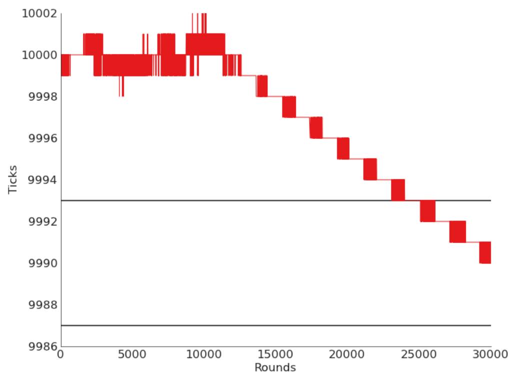
The four fitness measures defined in section 3.2.2 make it possible to specify the type of market behaviour that is favoured by the genetic algorithm. gives 16 combinatinos for how to optimize the model.

First of all, we are interested in establishing which parameters cause the market to return to a stable state after the fundamental price has incurred a shock

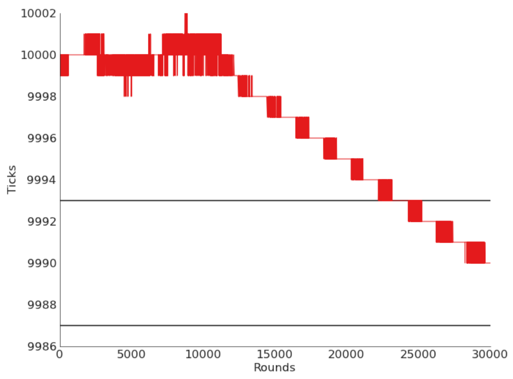
3.2.5 Filtering parameters

As mentioned earlier, it is not enough simply to define a fitness function which assigns high values to parameters causing realistic behavior. In addition, it is important to discard parameters which obviously do not correspond to a realistic setting. Imagine that the simulation scores high fitness values when executed without any market makers. Since it is known that real markets do in fact contain market makers, nothing can be inferred from such a result. Indeed this might be a consequence of poorly designed fitness measures, but since it is easier to use domain specific knowledge to filter out the unrealistic parameters

FIGURE 3.2: Example of a simulation which is assigned fairly good fitness values, but which was executed with clearly unrealistic parameters: $\mathtt{n_{mm}} = \mathtt{n_{sc}} = 0$. The simulation reaches the new fundamental price fairly quickly without any undershoot, and stays within the stability margin. The only point where it scores badly is the standard deviation which is slightly high due to the fluctuating trade price.



(A) $\lambda_{m,\mu}$=49, $\lambda_{m,\sigma}$=8, $\lambda_{c,\mu}$=32, $\lambda_{c,\sigma}$=19, f_{o}=1.0, f_{s}=29210.0, f_{m}=23970.0, f_{σ}=0.482



(B) $\lambda_{m,\mu}$=7, $\lambda_{m,\sigma}$=17, $\lambda_{c,\mu}$=21, $\lambda_{c,\sigma}$=17, f_{o}=1.0, f_{s}=28229.0, f_{m}=23146.0, f_{σ}=0.5

3.2.6 Handling failed simulations

Some parameters cause the simulation to act in strange ways, and even crash in some cases. For instance, the the order book becomes empty, the simulation throws an exception and terminates. Similarly, if the best bid/askprices drops to zero, the simulation exits. As such

The most common odd phenomenon was the market

3.3 Applying the genetic algorithm

Applying the genetic algorithm to produce markets with desirable behavior turned out to be more difficult that one could have hoped for. First of all, the high computational costs was a hurdle.

Secondly, many of the data sets (see section 3.3.3) produced did not contain any useful information.

The model parameters do influence the fitness values as they control model behavior, but they are not directly weighted into the fitness-values. This means that even after the parameters of the genetic algorithm was properly tuned in such a way that higher-fitness individuals were produced, these individuals often turned out to be of no interest. Such individuals were discarded according to the filtering criteria described in section 3.2.5 would have to be discarded. An example of such a case is discussed in section **??**

3.3.1 Time complexity

The high time complexity stems from several factors

- Running a simulation for a given set of parameters required up to several minutes of computation on a single CPU core.
- Large number of model parameters increase the size of the search space. The more parameters Unfortunately there is no magic to the way the genetic algorithm works, and optimizing in a larger search means a larger time complexity.
- Large range of parameters. Some of the parameters are integers while some are real numbers. While most of the parameters have a lower bound, none of the parameters have upper bounds.

- Unstable fitness parameters. Since the same set of parameters can produce varying model behavior, the fitness-values may also vary. Therefore it is necessary to evaluate the simulation several times for each set of parameters.
- The model parameters also influence the time complexity. For instance, evaluating a simulation with many agents takes more time than evaluating a simulation with fewer agents.

Genetic algorithms are naturally suited for parallel computation. Two servers with a total of 40 cores and enough memory to evaluate as many simulations were utilized. With this equipment, evaluating a single strategy (see section 3.3.2)for generating data sets could take

reaching a point where the genetic algorithm began to produce useful output turned out to be somewhat of an iterative process.

First of all, the computational cost of running the genetic algorithm was high, which meant that the algorithm was not always able to find better individuals.

Det tager lang tid -¿ faerre params skod resultater -¿ nye experimenter

Second of all, even when the genetic algorithm did manage to improve the fitness of the population, this did not always result in useful data.

3.3.2 Experiments: Dividing the search into parts

As mentioned earlier, the number of parameters and the range of each parameter influences the complexity of the search. Because of this, it is desirable to keep the number of parameters that are included in each individual as small as possible. However, fixing parameters means that some interesting properties about the model might not be discovered. Furthermore, varying all parameters at the same time makes the analysis and interpretation of the results more difficult. In an attempt to overcome this dilemma, several "experiments" were carried out [2]. Instead of trying to optimize all the model parameters at once, the search was split into several parts, each of which we call an experiment. Each of these experiments produce a data set, each of which were analyzed using the methods described in section 3.4. Some of the data sets produced interesting results, while others did not. Chapter **??** focuses on the analysis and presents the findings. A brief overview of the experiments is presented in **??**, but since the motivation for creating each data set is best understood in the context of the analysis of each data

[2]The reason for the quotes is that the term experiment might be stretching the common understanding of what an experiment is a little.

	$\lambda_{c,\mu}$	$\lambda_{c,\sigma}$	N_c	$T_{c,\mu}$	$T_{c,\sigma}$	$H_{c,\mu}$	$H_{c,\sigma}$	$W_{c,\mu}$	$W_{c,\sigma}$	$\lambda_{m,\mu}$	$\lambda_{m,\sigma}$	N_m	$T_{m,\mu}$	$T_{m,\sigma}$
0	84	11	14	98	9	1071	445	38	17	3	2	48	8	1
1	23	21	74	49	24	529	554	45	13	9	0	8	5	3
2	51	13	53	47	13	3586	536	10	11	9	4	14	4	2
3	18	21	213	70	39	793	1179	33	15	7	2	43	6	3
4	94	41	144	10	25	2668	893	12	15	6	1	49	7	4
5	19	4	130	15	38	1085	1165	39	4	2	3	11	4	4
6	65	15	91	81	46	3867	1991	48	1	7	2	21	4	4
7	36	38	143	77	19	2805	1870	10	9	7	0	3	2	4
8	43	8	10	19	19	3384	1706	33	4	8	4	5	5	0
9	11	33	127	94	49	3597	723	12	2	7	1	33	5	4

TABLE 3.3: An example data matrix containing the parameters of ten individuals who lived sometime during the execution of the genetic algorithm. In this case, each individual contained parameters for the number of HFT agents, as well as the latency and thinking time parameters. Hence, the data matrix has a column for each parameter.

set, the details are deferred until chapter **??**. The next section will explain exactly what a data set is.

3.3.3 Gene pool as data set

The previous sections contain the details of each of the steps undertaken in order to produce data sets. To summarize, the list below enumerates the steps.

1. Initialize a population in the genetic algorithm with healthy individuals.
2. Evaluate the fitness for every individual several times and obtain fitness-values by calculating averages.
3. Stack all individuals that ever lived into a $N \times \mathcal{L}_i$ parameter data matrix $\boldsymbol{P}$, where N is the number of individuals, and $\mathcal{L}_i$ is the length of each individual. Likewise, stack the fitness values into a $N \times \mathcal{L}_f$ fitness-data matrix $\boldsymbol{F}$, where $\mathcal{L}_f$is the number of fitness values calculated.
4. Filter the data by removing rows in $\boldsymbol{P}$with parameters which can be deemed not to correspond to real markets, and by removing rows in $\boldsymbol{F}$with fitness values that are not realistic. Please refer to section 3.2.5 for details. Naturally, when a row is removed in $\boldsymbol{P}$, it is also removed in $\boldsymbol{F}$, and vice versa.
5. Likewise, data points which were generated by a simulation crashing before it could complete were removed.

Tables 3.3 and 3.4 contain the first rows of $\boldsymbol{P}$and $\boldsymbol{F}$for one of the data sets.

	f_{o}	f_{m}	f_{σ}	f_{s}
0	3	25359	0.382092	29838
1	7	99999	1.289659	23373
2	6	99999	1.253363	18748
3	7	99997	1.695150	22819
4	6	94343	1.329276	22703
5	16	99999	2.439084	31860
6	6	93378	1.287235	25645
7	10	99997	1.858166	19417
8	3	24039	0.935465	27381
9	19	99995	4.092439	24845

Table 3.4: This table contains the fitness values for each individual in table 3.3. Note that, in order to increase the reliability of the fitness measure of an individual, the recorded fitness-values are the average of the fitness-values obtained by evaluating each individual ten times

3.3.4 Convergence of parameters

3.4 Data analysis tools

Using inverse simulation merely creates a lot of data. This data has to be analyzed before any

Data normalization

3.4.1 Data visualization

It is useful to be able to plot the data points

3.4.1.1 Color toned scatter plots

Scatter plots are useful for initial data analysis, as one can quickly detect problems such as outliers, and maybe even detect clusters of data points.

3.4.2 Preprocessing

3.4.2.1 Handling outliers

The term outliers is often used as a label for data which is considered “invalid” in the sense that is not a product of the true data generation process, but due to various sources of noise. In this report, data that was caused by failing simulations corresponds to the usual understanding of outliers. However, as was already explained in section 3.2.6, data from such simulations are never included in $\boldsymbol{P}$ and $\boldsymbol{F}$ to begin with. Instead, outliers in this report refer to entries in $\boldsymbol{P}$ and $\boldsymbol{F}$ deviate significantly from the majority of the data points by having extreme values.

Such data points caused problems when applying data analysis techniques which rely on a fairly normal distribution of the data points, such as Principal Component Analysis (PCA). PCA looks for a rotation of the data space, such that the axes of the new basis are aligned with the directions of the largest variance in the original space. Since a few points with extreme values come to account for a large portion of the data set variance, the new basis computed by PCA will be aligned along these few data points. When PCA is used to extract lower dimensional features from the data set, outliers will degrade the quality of these features. In the case that PCA is used for data visualization, the scatter plots of the first few principal components will not be very informative, as they merely show the projection of the data onto the axes aligned with the outliers.

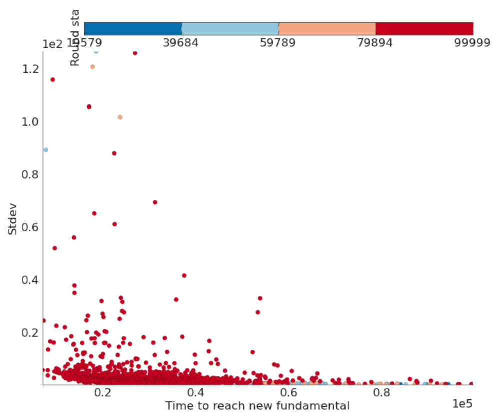
In all experiments, outliers were present in both the parameters space and in the fitness space. Outliers in the parameter space occur because of abnormally large mutations, and any dimension of the parameter space is susceptible to such an event. In the fitness space, only a few of the features suffered from the occurrence of outliers. The feature f_{m}cannot contain outliers, because the shock to the fundamental occurs at round $r = 10^4$, and because the simulation is terminated at round $r = 10^5$, hence $f_{\mathrm{m}} \in [10^4, 10^4 + 1, \ldots 10^5]$. The same is true for f_{s}. $f_\sigma$and f_{o}, on the other hand, are susceptible to outliers, because the two features have no upper bound.

1. Apply a monotonically increasing transformation $f(x)$ to some or all of the features for every data point in the data set. A common choice of f is $f(x) = \log x$, as it efficiently reduces the impact of data points with extremely high values. Figure ?? illustrates the effect of applying the log-transform. The log-scaling does a fairly good job of reducing the importance of the outliers in the $f_\sigma$feature, while the change is less dramatic in f_{s}. While the left scatter plot does not really reveal any structure of the data, the transformation makes it possible to spot to rough clusters when inspecting the right plot.

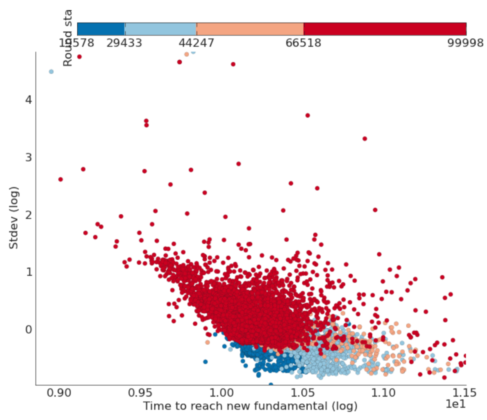
   Another simple method is to manually remove

2. Manually select one or more criteria for when a data point is to be considered an outlier. It is worth noticing that $f_{\mathrm{o}}$and f_σ

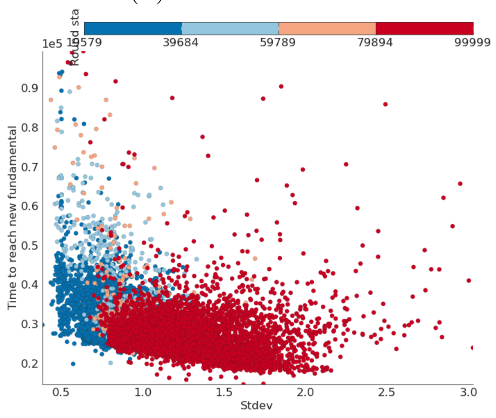
3. Use a one-class support vector machine to calculate a probability for each data point that said point is an inlier or an outlier. In this case, inliers



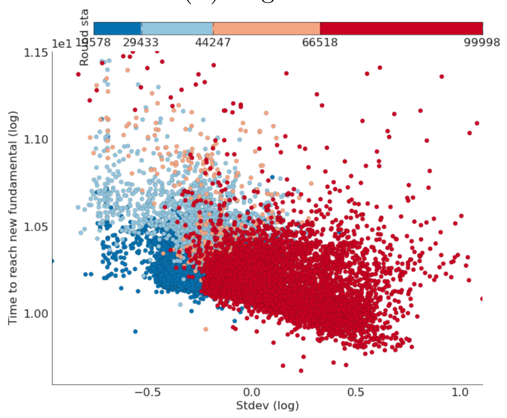
(C) Untouched data



(D) Log-scaled



(E) Without outliers



(F) Log-scaled and without outliers

FIGURE 3.3: Color toned scatter plots of f_{s}, $f_{\sigma}$and f_{m}taken from dataset d1after applying log-scaling and manually removing outliers

In the parameters data set, outliers can occur due to abnormally large mutations. Some parameters cause the simulation to act in strange ways, and even crash in some cases. For instance, the the order book becomes empty, the simulation throws an exception and terminates. Similarly, if the best bid/askprices drops to zero, the simulation exits. As such

The most common odd phenomenon was the market

3.4.2.2 title

3.4.3 Clustering algorithms

Chapter 4

Experiments

As mentioned in the previous chapter, the process of finding was an iterative one of running an experiment, analyzing the generated data, draw conclusions and then repeat the steps with a new experiments designed to amend the mistakes of the previous experiment. This chapter will go through each of these steps and explain the insights that were gained.

Data set d1was the first run of the genetic algorithm that actually produced something that looked like results. The following parameters were included in the genetic algorithm individuals:

$$\lambda_{c,\mu},\ \lambda_{c,\sigma},\ N_c,\ T_{c,\mu},\ T_{c,\sigma},\ H_{c,\mu},\ H_{c,\sigma},\ W_{c,\mu},\ W_{c,\sigma},\ \lambda_{m,\mu},\ \lambda_{m,\sigma},\ N_m,\ T_{m,\mu},\ T_{m,\sigma}$$

4.1 D3

In this experiment, the number of $N_m$and $N_c$as well all the latency parameters were included in the individuals. The genetic algorithm was run for 1000 generations with a population size of 200. A total of

4.2 D9

4.2.1 Calculating statistics for outliers

Dataset id	Parameters in genes
d1	$\lambda_{c,\mu}$, $\lambda_{c,\sigma}$, N_c, $T_{c,\mu}$, $T_{c,\sigma}$, $H_{c,\mu}$, $H_{c,\sigma}$, $W_{c,\mu}$, $W_{c,\sigma}$, $\lambda_{m,\mu}$, $\lambda_{m,\sigma}$, N_m, $T_{m,\mu}$, $T_{m,\sigma}$

	f_o¿ 10 (mean)	f_o¿ 10 (mean)	f_o¿ 10 (std)	f_o¿ 10 (std)
$\lambda_{c,\mu}$	73.8	21.2	21.2	13.2
$\lambda_{c,\sigma}$	5.2	9.6	9.6	5.7
$T_{c,\mu}$	64.3	24.3	24.3	13.6
$T_{c,\sigma}$	11.4	10.2	10.2	5.6
$H_{c,\mu}$	1804.5	2134.1	2134.1	1444.1
$H_{c,\sigma}$	1413.2	1024.0	1024.0	629.2
$W_{c,\mu}$	29.9	23.8	23.8	12.9
$W_{c,\sigma}$	3.6	9.5	9.5	5.8
$\lambda_{m,\mu}$	46.0	29.9	29.9	14.7
$\lambda_{m,\sigma}$	4.6	8.8	8.8	5.0
$T_{m,\mu}$	37.6	27.0	27.0	14.2
$T_{m,\sigma}$	0.9	10.0	10.0	6.0

Table 4.1: CAPTION

Appendix A

Additional tables

A.1 Dataset 1

Write your Appendix content here.

Appendix B

Third party software

Deap scoop sklearn matplotlib numpy, scipy and pandas geometric brownian walk

Bibliography

[1] C. J. Hawthorn, K. P. Weber, and R. E. Scholten. Littrow configuration tunable external cavity diode laser with fixed direction output beam. *Review of Scientific Instruments*, 72(12):4477–4479, December 2001. URL `http://link.aip.org/link/?RSI/72/4477/1`.

Bibliography

[1] Chi Wang, Kiyoshi Izumi, Takanobu Mizuta and Shinobu Yoshimura: "Investigating the Impact of Trading Frequencies of Market Makers: a Multi-agent Simulation Approach" *SICE Journal of Control Measurement, and System Integration, Vol. 4, No. 1, pp. 001-005, January 2011.*

[2] Michael J. McGowan: "The Rise of Computerized High Frequency Trading: Use and Controversies", *2010 Duke L. & Tech. Rev., 2010.*

[3] Thomas McInish and James Upson: "Strategic Liquidity Supply in a Market with Fast and Slow Traders", *Available at SSRN 1924991 (2012).*

[4] Niel Johnson, Guannan Zhao, Eric Hunsader, Jing Meng, Amith Ravindar, Spencer Carran and Brian Tivnan: "Financial Black Swans Driven by Ultrafast Machine Ecology", *Available at SSRN (2012).*

[5] Kiyoshii Izumi, Kiyoshi, Fujio Toriumi, and Hiroki Matsui: "Evaluation of automated-trading strategies using an artificial market", *Neurocomputing 72.16 (2009): 3469-3476.*

[6] Chiarella, Carl, Giulia Iori, and Josep Perelló: "The impact of heterogeneous trading rules on the limit order book and order flows" *Journal of Economic Dynamics and Control 33.3 (2009): 525-537.*

[7] Peter Gomber, Björn Arndt, Marco Lutat, Tim Uhle: "High-frequency trading." *Available at SSRN 1858626 (2011).*

[8] J. Doyne Farmer and Spyros Skouras: "An ecological perspective on the future of computer trading" *Quantitative Finance 13.3 (2013): 325-346.*