

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

<b>Declaration of Authorship</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>viii</b>
<b>Abbreviations</b>	<b>ix</b>
<b>Physical Constants</b>	<b>x</b>
<b>Physical Constants</b>	<b>xi</b>
<b>Symbols</b>	<b>xii</b>
<b>1 Model</b>	<b>2</b>
1.1 Model . . . . .	2
1.1.1 Asynchronous vs game theoretic model . . . . .	2
1.1.2 Overall architecture . . . . .	2
1.1.3 Modeling delays . . . . .	3
1.2 Market components . . . . .	3
1.2.1 Stocks . . . . .	3
1.2.1.1 Fundamental price . . . . .	4
1.2.2 Messages . . . . .	4
1.2.2.1 Market information . . . . .	5
1.2.2.2 Orders . . . . .	6
1.2.2.3 Transaction receipts . . . . .	6
1.2.2.4 Order cancellations . . . . .	7
1.2.3 Order book . . . . .	7
1.2.3.1 Price updating . . . . .	7

1.2.3.2	Order matching	9
1.2.3.3	Empty order book	9
1.2.4	Market	9
1.2.4.1	Short selling	9
1.3	Agents	10
1.3.1	Slow traders	11
1.3.1.1	Arrival of orders	13
1.3.2	Market makers	13
1.3.3	HFT Chartists	14
1.4	Simulation rounds	14
1.5	Implementation	14
1.6	Experiments	14
<b>2</b>	<b>Parameter tuning</b>	<b>15</b>
2.1	Motivation and overall procedure	15
2.1.1	Selecting fixed parameters	17
2.1.2	Filtering parameters	17
2.2	Inverse simulation with a genetic algorithm	17
2.2.1	Representing parameters as genes	18
2.2.2	Model fitness	19
2.3	Controlling market behavior	19
2.4	Experiments	19
2.5	Data sets and analysis	20
2.6	Data analysis	21
2.6.1	Preprocessing	21
2.6.1.1	Removing outliers	21
2.6.2	Data visualization	22
2.6.2.1	Color toned scatter plots	22
2.6.2.2	title	22
2.6.3	Clustering algorithms	22
<b>3</b>	<b>Data Analysis</b>	<b>23</b>
3.1	Dataset 1	23
3.1.1	Other decompositions	23
<b>A</b>	<b>Dataset statistics</b>	<b>26</b>
A.1	Dataset 1	26
	<b>Bibliography</b>	<b>28</b>

# List of Figures

1.1	:	4
1.2		5
1.3	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.	6
1.4		8
1.5	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.	10
1.6	Slow trader order arrival	13
2.1	<b>Motivatoin for tuning:</b> the two	16
2.2	Example of a simulation which is assigned fairly good fitness values, but which was executed with clearly unrealistic parameters: $n_{mm} = n_{sc} = 0$ . 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.	18
2.3		20



# List of Tables

1.1	TABLE ILLUSTRATING ORDER BOOK . . . . .	7
2.1	Overview of parameters used in the genetic algorithm . . . . .	18
2.2	Overview of experiments . . . . .	19
2.3	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 paremeters for the number of HFT agents, as well as the latency and thinking time parameters. Hence, the data matrix has a column for each. . . . .	21
2.4	This table contains the fitness values for each individual in table 2.3. 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 . . . . .	21
3.1	Mean for parameters and fitness values for each cluster in the parameter space for dataset d3. . . . .	24
3.2	Std for parameters and fitness values for each cluster in the parameter space for dataset d3. . . . .	24
3.3	Mean for parameters and fitness values for each cluster in the parameter space for dataset d3. . . . .	25
3.4	Std for parameters and fitness values for each cluster in the parameter space for dataset d3. . . . .	25
A.1	Median for parameters and fitness values for each cluster in the parameter space for dataset d3. . . . .	26
A.2	Min for parameters and fitness values for each cluster in the parameter space for dataset d3. . . . .	27
A.3	Max for parameters and fitness values for each cluster in the parameter space for dataset d3. . . . .	27

# Abbreviations

**HFT**   **H**igh **F**requency **T**rader

**ST**   **S**low **T**rader

**OB**   **O**rders **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 buy order happen to have the same listed price, they are said to match.
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 $\tau_{i,j} = \tau_{j,i}$ .
$s_t$	Spread at the end of round $t$
$p_t^a$	The lowest askprice in the order book at the end of round $t$
$p_t^b$	The highest bidprice in the order book at the end of round $t$
$p^m$	Match price, i.e., the price at which a trade is executed
$f_t$	Fundamental price at round $t$

# Symbols

Symbol	Description
$n_{\text{rounds}}$	Number of simulation rounds
$***\lambda$	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.
$n_{\text{mm}}$	Number of hft market maker agents.
$n_{\text{sc}}$	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



# Chapter 1

## Model

### 1.1 Model

As explained in section ??, 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.

#### 1.1.1 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.

#### 1.1.2 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.

### 1.1.3 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

## 1.2 Market components

Sending/receiving orders, supply liquidity

### 1.2.1 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 it is 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



FIGURE 1.1: :

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.

#### 1.2.1.1 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

#### 1.2.2 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 1.2

#### 1.2.2.1 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 1.3 summarizes the procedure.

This is analogous to how



---

FIGURE 1.3: 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.

### 1.2.2.2 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 agent 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 [1.3](#).

### 1.2.2.3 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 1.1: TABLE ILLUSTRATING ORDER BOOK

delay, the agents do 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.

#### 1.2.2.4 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 1.3.2. 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.

#### 1.2.3 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 simulation be a non-negative price difference between the sell order with the lowest price and the buy order with the highest price. This difference is called the *spread*, and is denoted as follows

$$s_t = p_t^a - p_t^b \quad (1.1)$$

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

When an agent

##### 1.2.3.1 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 1.1 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 1.4




---

FIGURE 1.4

When no orders arrive or are removed from the order book, the prices are updated as  $p_{t+1}^a = p_t^a$  and  $p_{t+1}^b = p_t^b$ .

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  $p_t^a$  and  $p_t^b$ . Messages are therefore processed in random orders, so that no agent is favored.

### 1.2.3.2 Order matching

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

$$\Delta v = \min(v_{o_1}, v_{o_2})$$

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  $\Delta v$ . 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.

### 1.2.3.3 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

## 1.2.4 Market

### 1.2.4.1 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 1.5. 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 1.5 may seem unlikely, it did in fact occur frequently, making it necessary to implement handling of this special case.



---

FIGURE 1.5: 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.

### 1.3 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]

### 1.3.1 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 1.2.1.1). 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<sup>1</sup>. 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 1.3.3.

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 1.6.

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

---

<sup>1</sup>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.

#### 1.3.1.1 Arrival of orders



---

FIGURE 1.6: Slow trader order arrive randomly according to a Poisson process. The bars show the volume generated by slow traders. XXXWRITEMORE

#### 1.3.2 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.

### **1.3.3 HFT Chartists**

## **1.4 Simulation rounds**

## **1.5 Implementation**

## **1.6 Experiments**

WRITE ABOUT KEY POINTS OF INTEREST IN TIME

## Chapter 2

# Parameter tuning

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

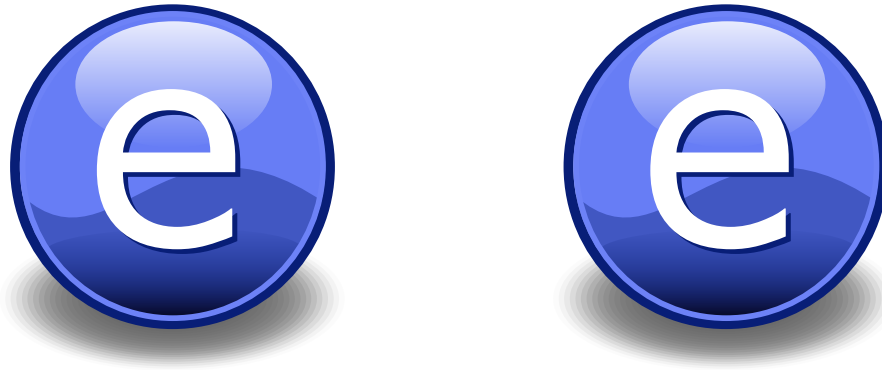
The parameter tuning turned out to consume a significant amount of time, and simple using a genetic algorithm to optimize over the entire space of parameters was not enough. Instead, the process was a slow and iterative one of running the genetic algorithm to create a data set, analyze the data set to 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.

### 2.1 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 [2.1a](#). 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 [2.2.2](#). 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 2.1: **Motivatoin for tuning:** the two

but do not correspond to a realistic market setting. An example for this is given in figure 2.1b, and section 2.1.2 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 2.2.2

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<sup>1</sup>. 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 2.2.
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.

<sup>1</sup>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.

### 2.1.1 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

**n<sub>rounds</sub>** 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.

### 2.1.2 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

## 2.2 Inverse simulation with a genetic algorithm

A basic type of genetic algorithm was used. The algorithm proceeds as explain 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  $n_{\text{gen}}$  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.

Table 2.1 presents an overview of the parameters used in the genetic algorithm.





FIGURE 2.2: Example of a simulation which is assigned fairly good fitness values, but which was executed with clearly unrealistic parameters:  $\mathbf{n}_{\text{mm}} = \mathbf{n}_{\text{sc}} = 0$ . 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.

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 2.1: Overview of parameters used in the genetic algorithm

### 2.2.1 Representing parameters as genes

A set of parameters is represented by an individual,  $i$ , consisting of one floating point  $g_{i,j}$  number for each parameter, 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  $g_j \in [0; 1]$ :

$$g_{i,j} \mathcal{U}(0, 1) \tag{2.1}$$

Most of the parameters in the model are integers

Parameter	Range
$n_{mm}$	
$n_{sc}$	

TABLE 2.2: Overview of experiments

### 2.2.2 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

- 
- 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.

## 2.3 Controlling market behavior

Inverse simulation is a handy tool, because it makes it possible to specify desired behavior, and then search for parameters yielding such behavior. The four fitness measures defined in section 2.2.2 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

## 2.4 Experiments

This section will outline the varioius experiments carried out with the fundamental.



FIGURE 2.3

## 2.5 Data sets and analysis

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  $\mathbf{P}$ , 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  $\mathbf{F}$ , where  $\mathcal{L}_f$  is the number of fitness values calculated.
4. Filter the data by removing rows in  $\mathbf{P}$  with parameters which can be deemed not to correspond to real markets, and by removing rows in  $\mathbf{F}$  with fitness values that are not realistic. Please refer to section 2.1.2 for details. Naturally, when a row is removed in  $\mathbf{P}$ , it is also removed in  $\mathbf{F}$ , and vice versa.
- 5.

	$\lambda_{c,\mu}$	$\lambda_{c,\sigma}$	$N_c$	$T_{c,\mu}$	$T_{c,\sigma}$	$H_{c,\mu}$	$H_{c,\sigma}$	$W_{c,\mu}$	$W_{c,\sigma}$	$\lambda_{m,\mu}$	$\lambda_{m,\sigma}$	$N_m$	$T_{m,\mu}$	$T_{m,\sigma}$
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 2.3: 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 paremeters for the number of HFT agents, as well as the latency and thinking time parameters. Hence, the data matrix has a column for each.

	$f_{\text{overshoot}}$	$f_{\text{stable}}$	$f_{\text{stdev}}$	$f_{\text{speed}}$
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 2.4: This table contains the fitness values for each individual in table 2.3. 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

## 2.6 Data analysis

### 2.6.1 Preprocessing

Data normalization

#### 2.6.1.1 Removing outliers

Outliers, that is data points which deviate significantly from the majority of the data points, can cause problems when applying data analysis techniques which rely on a fairly normal distribution of the data points, such as Principal Component Analysis.

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

## **2.6.2 Data visualization**

It is useful to be able to plot the data points

### **2.6.2.1 Color toned scatter plots**

#### **2.6.2.2 title**

### **2.6.3 Clustering algorithms**

## Chapter 3

# Data Analysis

Take the best genes and simulate several times. Show bar plot of each gene, where the heights of the bars are the mean fitness, and with some brackets showing std of fitness.

### 3.1 Dataset 1

As a first step in the data analysis, it is useful to visualize the dataset. Figure

The first attempt at PCA: The hope is that some regions in the parameter space correspond to regions in the fitness space.

#### 3.1.1 Other decompositions

	c0	c1	c2	c3	c4	c5	c6	c7
$\lambda_{c,\mu}$	95.6	91.6	61.1	81.2	86.0	76.4	67.6	70.6
$\lambda_{c,\sigma}$	3.5	4.4	24.0	10.3	5.0	15.4	20.6	20.0
$N_c$	7.0	9.1	61.4	15.5	6.5	33.5	44.7	39.4
$T_{c,\mu}$	78.7	76.9	54.7	54.8	79.4	57.2	53.4	61.8
$T_{c,\sigma}$	16.0	15.0	29.4	22.9	26.0	25.8	34.7	29.2
$H_{c,\mu}$	3282.9	3827.9	167.6	3332.1	4580.1	1625.1	1629.4	4335.4
$H_{c,\sigma}$	74.7	309.2	1078.0	1697.9	152.9	1757.3	592.5	1521.9
$W_{c,\mu}$	18.3	17.0	32.2	29.0	14.4	31.1	32.3	30.5
$W_{c,\sigma}$	6.3	5.8	6.4	5.2	6.7	6.0	7.4	6.3
$\lambda_{m,\mu}$	5.8	3.7	5.7	7.5	4.0	6.2	6.4	6.7
$\lambda_{m,\sigma}$	2.7	3.8	2.3	2.5	4.0	2.3	2.1	2.4
$N_m$	0.4	0.6	5.0	2.4	0.5	3.3	4.7	3.9
$T_{m,\mu}$	5.2	4.0	5.4	5.7	4.2	6.1	6.1	5.7
$T_{m,\sigma}$	1.6	3.0	2.3	1.5	2.8	1.8	2.1	1.9
$f_{\text{overshoot}}$	2.4	2.6	6.4	3.0	2.4	4.2	4.9	5.2
$f_{\text{stable}}$	27236.4	29936.2	60043.7	33472.2	27105.4	45194.7	50751.5	43294.2
$f_{\text{stddev}}$	0.7	0.8	1.4	0.8	0.7	1.0	1.2	1.2
$f_{\text{speed}}$	27411.6	27416.0	27437.0	28332.8	27438.2	27760.4	28080.6	27684.4

TABLE 3.1: Mean for parameters and fitness values for each cluster in the parameter space for dataset d3.

	c0	c1	c2	c3	c4	c5	c6	c7
$\lambda_{c,\mu}$	14.1	13.9	30.3	23.6	16.7	23.0	29.4	28.2
$\lambda_{c,\sigma}$	5.2	6.4	11.0	12.6	6.9	15.1	13.5	16.0
$N_c$	22.0	23.4	70.2	36.4	18.4	53.9	60.3	68.3
$T_{c,\mu}$	20.7	16.8	27.5	29.1	18.9	23.7	26.1	25.3
$T_{c,\sigma}$	17.2	16.2	16.5	19.0	17.5	17.5	18.3	17.1
$H_{c,\mu}$	224.1	174.9	215.8	192.3	356.3	415.8	327.3	397.8
$H_{c,\sigma}$	145.6	265.7	548.6	138.4	181.4	210.0	374.5	326.4
$W_{c,\mu}$	8.0	10.8	13.7	11.1	6.1	14.0	15.2	13.1
$W_{c,\sigma}$	1.8	2.6	5.6	2.3	1.9	3.6	5.3	3.6
$\lambda_{m,\mu}$	4.0	3.6	2.7	2.8	3.4	2.8	2.7	2.7
$\lambda_{m,\sigma}$	2.4	2.2	1.6	1.9	2.5	1.8	1.8	1.8
$N_m$	2.5	2.2	6.1	3.4	2.4	4.6	6.4	6.4
$T_{m,\mu}$	1.9	1.9	2.8	1.8	2.0	2.2	2.0	2.1
$T_{m,\sigma}$	1.7	1.4	1.6	1.6	1.4	1.4	1.6	1.6
$f_{\text{overshoot}}$	3.9	4.0	11.7	2.9	4.9	8.4	11.1	14.5
$f_{\text{stable}}$	16318.8	20471.9	36132.0	23560.8	15722.1	32640.3	34726.7	32005.2
$f_{\text{stddev}}$	0.9	1.0	2.7	0.6	1.1	2.1	3.1	3.6
$f_{\text{speed}}$	2500.3	2316.3	6059.9	4255.1	2067.0	4774.2	6205.2	4767.4

TABLE 3.2: Std for parameters and fitness values for each cluster in the parameter space for dataset d3.

	c0	c1	c2	c3	c4	c5	c6	c7
$\lambda_{c,\mu}$	57.5	72.1	77.2	72.5	75.1	91.3	68.1	64.7
$\lambda_{c,\sigma}$	22.0	16.8	11.8	17.4	14.8	5.5	18.7	23.9
$N_c$	124.3	25.6	24.8	43.1	18.2	4.6	91.7	62.4
$T_{c,\mu}$	53.9	55.7	61.8	59.5	55.8	71.4	56.5	62.1
$T_{c,\sigma}$	24.0	29.0	22.5	26.0	26.6	18.9	22.5	21.9
$H_{c,\mu}$	2674.5	2930.3	3129.6	2785.0	3043.2	3549.4	2711.3	2955.4
$H_{c,\sigma}$	991.9	1116.8	853.8	970.8	1058.8	541.5	951.4	1028.4
$W_{c,\mu}$	26.3	29.7	24.5	26.3	26.6	20.1	23.4	24.9
$W_{c,\sigma}$	8.7	6.7	6.2	7.3	5.9	5.9	7.1	6.9
$\lambda_{m,\mu}$	5.1	6.7	5.1	5.7	6.4	5.0	5.1	5.4
$\lambda_{m,\sigma}$	2.7	2.7	3.3	2.8	3.1	3.2	2.6	3.2
$N_m$	13.8	7.6	3.2	9.6	5.0	0.6	5.3	12.2
$T_{m,\mu}$	5.2	5.2	4.8	4.7	5.0	4.7	5.2	3.7
$T_{m,\sigma}$	2.6	1.8	2.6	2.0	2.3	2.3	2.4	2.1
$f_{\text{overshoot}}$	10.7	2.7	4.8	2.5	5.0	2.2	8.6	4.1
$f_{\text{stable}}$	98091.5	34766.3	79748.1	48406.7	57603.6	23436.6	97890.2	86745.7
$f_{\text{stddev}}$	2.5	0.8	1.1	0.7	1.1	0.7	1.7	1.0
$f_{\text{speed}}$	43040.2	38863.7	27603.0	61632.1	30519.9	27566.4	24705.4	82134.1

TABLE 3.3: Mean for parameters and fitness values for each cluster in the parameter space for dataset d3.

	c0	c1	c2	c3	c4	c5	c6	c7
$\lambda_{c,\mu}$	30.2	24.0	27.2	26.1	23.1	17.2	31.1	29.0
$\lambda_{c,\sigma}$	15.2	14.1	13.1	15.5	13.9	8.4	15.9	16.3
$N_c$	96.4	32.8	23.9	56.9	24.7	5.9	70.5	88.9
$T_{c,\mu}$	29.0	27.2	27.7	26.7	28.2	23.5	28.6	31.1
$T_{c,\sigma}$	17.2	18.4	18.8	19.0	18.4	18.4	18.0	17.1
$H_{c,\mu}$	1386.1	1199.3	1228.1	1177.5	1239.2	780.5	1442.0	1015.7
$H_{c,\sigma}$	590.4	665.2	676.3	697.1	706.8	645.9	655.3	669.5
$W_{c,\mu}$	14.0	12.5	13.8	13.0	12.1	11.5	14.4	12.4
$W_{c,\sigma}$	5.2	4.1	3.6	4.2	3.2	2.3	4.9	3.7
$\lambda_{m,\mu}$	2.7	2.7	3.5	3.0	3.0	3.8	3.0	3.1
$\lambda_{m,\sigma}$	1.8	2.0	2.2	2.0	2.0	2.3	2.0	1.9
$N_m$	8.6	4.7	4.4	5.5	4.0	1.9	7.0	10.0
$T_{m,\mu}$	2.7	2.2	2.3	2.3	2.1	2.0	2.5	2.2
$T_{m,\sigma}$	1.4	1.7	1.6	1.7	1.6	1.7	1.6	1.7
$f_{\text{overshoot}}$	13.2	0.8	16.1	0.8	18.8	2.3	13.3	2.3
$f_{\text{stable}}$	4181.8	4493.8	5262.8	10233.5	6590.6	1176.6	3235.0	11181.6
$f_{\text{stddev}}$	3.7	0.2	3.9	0.2	5.1	0.6	3.4	0.6
$f_{\text{speed}}$	6771.8	5104.2	3004.9	9614.7	4646.2	1327.2	4273.9	9590.5

TABLE 3.4: Std for parameters and fitness values for each cluster in the parameter space for dataset d3.



# Appendix A

## Dataset statistics

### A.1 Dataset 1

Write your Appendix content here.

	c0	c1	c2	c3	c4	c5	c6	c7
$\lambda_{c,\mu}$	85.0	74.0	65.0	74.0	74.0	79.0	92.0	61.0
$\lambda_{c,\sigma}$	7.0	15.0	26.5	16.0	15.0	14.0	2.0	21.0
$N_c$	23.0	10.0	10.0	74.0	10.0	10.0	3.0	126.0
$T_{c,\mu}$	77.0	54.0	75.5	68.0	68.0	54.0	79.0	57.5
$T_{c,\sigma}$	16.0	41.0	19.0	19.0	24.0	24.0	12.0	20.0
$H_{c,\mu}$	3384.0	3384.0	3384.0	3384.0	3318.0	3384.0	3636.0	2855.0
$H_{c,\sigma}$	590.0	1325.0	938.0	835.0	962.0	1325.0	221.0	962.0
$W_{c,\mu}$	18.0	33.0	19.0	17.0	30.0	33.0	17.0	27.5
$W_{c,\sigma}$	7.0	6.0	7.0	7.0	7.0	5.0	7.0	8.0
$\lambda_{m,\mu}$	7.0	8.0	6.5	5.0	7.0	8.0	2.0	5.0
$\lambda_{m,\sigma}$	4.0	4.0	4.0	3.0	4.0	4.0	4.0	3.0
$N_m$	3.0	5.0	8.5	4.0	8.0	5.0	0.0	12.0
$T_{m,\mu}$	5.0	5.0	3.0	5.0	5.0	5.0	5.0	5.0
$T_{m,\sigma}$	3.0	1.0	2.0	3.0	2.0	3.0	3.0	3.0
$f_{\text{overshoot}}$	4.0	3.0	3.0	6.0	2.0	4.0	2.0	7.0
$f_{\text{stable}}$	79924.0	32827.0	86068.5	99898.0	47143.0	56431.0	23291.0	99995.0
$f_{\text{stddev}}$	0.9	0.8	0.8	1.3	0.7	0.9	0.7	1.6
$f_{\text{speed}}$	26814.0	38071.0	79619.5	24699.0	58541.0	29665.5	27368.0	41906.5

TABLE A.1: Median for parameters and fitness values for each cluster in the parameter space for dataset d3.

	c0	c1	c2	c3	c4	c5	c6	c7
$\lambda_{c,\mu}$	2.0	1.0	5.0	1.0	1.0	2.0	1.0	1.0
$\lambda_{c,\sigma}$	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
$N_c$	0.0	0.0	3.0	0.0	2.0	0.0	0.0	3.0
$T_{c,\mu}$	3.0	3.0	3.0	1.0	3.0	3.0	3.0	3.0
$T_{c,\sigma}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$H_{c,\mu}$	26.0	26.0	731.0	26.0	26.0	26.0	26.0	26.0
$H_{c,\sigma}$	0.0	0.0	8.0	0.0	8.0	8.0	0.0	8.0
$W_{c,\mu}$	1.0	2.0	4.0	1.0	2.0	2.0	1.0	2.0
$W_{c,\sigma}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\lambda_{m,\mu}$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\lambda_{m,\sigma}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$N_m$	0.0	1.0	4.0	0.0	2.0	0.0	0.0	2.0
$T_{m,\mu}$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$T_{m,\sigma}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$f_{\text{overshoot}}$	4.0	1.0	2.0	4.0	1.0	4.0	1.0	4.0
$f_{\text{stable}}$	66349.0	26837.0	63503.0	86360.0	25076.0	41941.0	19579.0	71013.0
$f_{\text{stddev}}$	0.7	0.4	0.4	0.7	0.5	0.6	0.4	0.7
$f_{\text{speed}}$	16968.0	27859.0	69382.0	6964.0	49219.0	17726.0	7681.0	35831.0

TABLE A.2: Min for parameters and fitness values for each cluster in the parameter space for dataset d3.

	c0	c1	c2	c3	c4	c5	c6	c7
$\lambda_{c,\mu}$	133.0	119.0	115.0	133.0	127.0	116.0	133.0	119.0
$\lambda_{c,\sigma}$	55.0	55.0	49.0	55.0	55.0	55.0	55.0	55.0
$N_c$	296.0	273.0	297.0	314.0	298.0	273.0	263.0	314.0
$T_{c,\mu}$	116.0	113.0	113.0	116.0	113.0	102.0	118.0	112.0
$T_{c,\sigma}$	56.0	61.0	50.0	61.0	50.0	55.0	61.0	49.0
$H_{c,\mu}$	6185.0	5255.0	4436.0	6185.0	4758.0	4969.0	6185.0	4850.0
$H_{c,\sigma}$	2416.0	2132.0	1960.0	2416.0	1991.0	2050.0	2248.0	1991.0
$W_{c,\mu}$	53.0	53.0	47.0	55.0	48.0	52.0	62.0	50.0
$W_{c,\sigma}$	19.0	19.0	18.0	19.0	19.0	19.0	19.0	19.0
$\lambda_{m,\mu}$	12.0	10.0	10.0	12.0	10.0	11.0	13.0	10.0
$\lambda_{m,\sigma}$	8.0	8.0	7.0	8.0	8.0	8.0	8.0	8.0
$N_m$	48.0	44.0	49.0	49.0	28.0	44.0	48.0	49.0
$T_{m,\mu}$	11.0	9.0	9.0	11.0	9.0	10.0	11.0	11.0
$T_{m,\sigma}$	5.0	5.0	4.0	5.0	5.0	5.0	5.0	5.0
$f_{\text{overshoot}}$	443.0	6.0	12.0	446.0	5.0	493.0	391.0	132.0
$f_{\text{stable}}$	87742.0	49188.0	99997.0	99999.0	70065.0	67674.0	41296.0	99999.0
$f_{\text{stddev}}$	105.6	1.3	2.5	126.1	1.2	126.7	89.4	41.7
$f_{\text{speed}}$	46840.0	56024.0	99534.0	37225.0	94272.0	49404.0	37070.0	66693.0

TABLE A.3: Max for parameters and fitness values for each cluster in the parameter space for dataset d3.

# 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 McNish 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.