UNIVERSITY OF OKLAHOMA

SCHOOL OF INDUSTRIAL ENGINEERING

ISE 5103 – INTELLIGENT DATA ANALYTICS

(*CHARLES NICHOLSON*)

MODELING LIMIT ORDER BOOK PRICE DYNAMICS

STUDENT X

FALL 2014

Table of Contents

1. EXECUTIVE SUMMARY

The limit order book contains a massive amount of rapidly evolving data, with the possibility of important patterns forming and dissolving at a frequency too high for a human observer. Hence, we would like to predict the occurrence and direction of spread crossing and provide trading strategies that would lead to profit. We considered sixty features for level 10 data obtained from [5]: best ask price, best ask volume, best bid price, best bid volume, mid-price, and spread for each level. Our main target was to predict the occurrence and direction of bid-ask spread crossing ("spread_price") after a number of events k. This happens rarely, but when it happens, it guarantees profit which is our goal for this project. Hence, our target had three possible values: "stat" for no spread crossing, "up" for an upward spread crossing and "down" for a downward spread crossing. The number of events considered in our model was 30. The implemented trading strategy based on the spread crossing direction at an event i is described here: do nothing if no spread ("stat"), buy at best ask at event i and sell at best bid at event $i + k$ if upward ("up"), and buy at best ask at event $i + k$ and sell at best bid at event i if downward ("down").

Applying the suggested trading strategy in this project, yield to making profit of $5,999 from trading 100 shares when the spread crossing is "up" and 1000 shares when it is "down". However, several parameters can affect the profit amount: the trading strategy, the number of events used in the model, and the number of shares to be traded.

As a future work, different predictive models could be built and compared with the suggested model using criteria such as accuracy and computation time. Also, the number of features can be limited by selecting the most important ones using techniques such as Information Gain. Moreover, since we could not produce any profit consistently using the predictive models for mid-price movement, we would like to explore if adding mid-price movement as a predictor would affect the performance of the current model. Also, the effectiveness of the suggested model could be tested on low frequency trading, so that people can use this for personal gain/profit.

2. HIGH FREQUENCY TRADING LIMIT ORDER BOOK MODEL

2.1 HIGH FREQUENCY TRADING (HFT)

High frequency trading (HFT) is high-volume trading used by proprietary traders and a new breed of electronic trading outfits (typically privately held) [1]. It processes trading information faster than other traders in the market, using sophisticated trading algorithms and powerful computers. Often the HFT trading systems are co-located with the exchanges' electronic matching systems within securities exchanges, which slightly (the time savings are in the milliseconds) speed up their orders to the exchanges and provide a competitive advantage over other traders whose systems are not as fast.

2.2 LIMIT ORDER BOOK (LOB)

In securities exchanges, an order book contains the list of interested buyers and the list of interested sellers. For each entry it must keep among others, some form of identifying the party, the number of shares and the price that the buyer or seller are bidding/asking for the particular security. [7]

2.3 MOTIVATION

The limit order book contains a massive amount of rapidly evolving data, with the possibility of important patterns forming and dissolving at a frequency too high for a human observer. Predicting the occurrence and direction of price movement and spread crossing can provide guidance to develop trading strategies that would lead to profit [3]. Therefore, we chose supervised machine learning techniques to predict direction of price movement and spread crossing.

2.4 DATA

Records of high-frequency trading activity are organized into a database with two major components, the "message" and the "orderbook" files [5]. (See Appendix A)

The "message" file stores basic information on each trading event, such as time of occurrence and transaction type, price, volume and direction. The "orderbook" contains limit orders information for bid and asks events. Each entry of the order book groups ask and bid events on n different price levels (we take n = 10) along with their volume sizes. The best ask and best bid are listed first, and the next best second, etc.

Table 1: sample of the "orderbook" and "message" file for AAPL (Apple Company)

	level 1				**level 2**				**level 3**				**…**
event	Ask price	Ask vol.	Bid price	Bid vol.	Ask price	Ask vol.	Bid price	Bid vol.	Ask price	Ask vol.	Bid price	Bid vol.	…
k-1	585.69	16	585.44	167	585.71	118	585.40	50	585.72	2	585.38	22	…
k	585.71	118	585.44	167	585.72	2	585.40	50	585.74	18	585.38	22	…
...	…	…	…	…	…	…	…	…	…	…	…	…	…
k+4	585.71	118	585.70	66	585.72	2	585.44	167	585.75	4	585.40	50	…
k+5	585.71	118	585.70	66	585.72	2	585.44	167	585.80	100	585.40	50	…
...	…	…	…	…	…	…	…	…	…	…	…	…	…
k+8	585.71	100	585.44	167	585.80	100	585.40	50	585.48	100	585.38	22	…
k+9	585.71	100	585.45	18	585.80	100	585.44	167	585.81	100	585.40	50	…
k+10	585.68	18	585.45	18	585.71	100	585.44	167	585.80	100	585.40	50	…

From Table 1 above, we observe that a new entry in the message file typically causes one fresh record to be added into the order book.

For instance, the transaction event at the k-th row of the message file of Table 1,

- Execution of an ask order at the price $585.69 with 16 shares,
- Exactly cancels out the best ask price and its volume in Row k-1 of the order book,
- Making the next best ask price, $585.71, become the new best ask price as shown in Row k of the order book.
- It can also be observed from the message file that multiple trading events could arrive within milliseconds as demonstrated from Row k-1 to k+10, leading to drastic fluctuation of prices and volumes in the order book.

Although a variety of "metrics" have been designed to capture the price fluctuation, in this project we select as metrics:

1. The occurrence and direction of mid-price movement, and
2. The occurrence and direction of bid-ask spread crossing.

3. THE MODEL

3.1 PROBLEM DESCRIPTION

In HFT, we are looking for ways to capture patterns in the data and predict those patterns in future instances of data, in order to make profit. The LOB has many levels corresponding to queues where orders (prices and matching volumes) queue up to be executed. The incoming orders are ranked depending on the price, and then if two orders enter the queue at the same price level, first in first out (FIFO) rule is applied. Two patterns are interesting: the mid-price movement and the spread crossing.

The mid-price movement is the direction of the mid-price which could be an indicator of a possible profit if detected accurately. But this will not guarantee making profit. The best ask and bid prices are particularly interesting. Using them is the only way to make a transaction happen. Placing a limit order does not guarantee its execution. On the other hand, placing a market order will guarantee the execution of an order. A market sell order is placed at the highest bid and a market buy order is placed at the highest ask.

The spread crossing detects when the best ask and bid prices cross at different points in times or after a set number of events. This happens rarely, but when it happens, it guarantees profit because buying at best Bid and selling at best Ask (market orders) when a spread crossing happens will guarantee profit.

Therefore, performing predictive modeling that accurately predicts when these spread crossings happen is extremely critical to make profit. Hence, we focus on the spread crossing as a target for this project.

3.2 SOLUTION TECHNIQUE

We first tried several techniques to detect when spread crossing happens. We calculated the spread crossings happening within a one second and a two second range. We were able to count few cases where the spread crossing happens. But, this did not allow us to consistently and accurately find the points in time or events when we have a spread crossing. Hence, we based our spread crossing detection method on the number of events k that separate two best bid and

ask prices. We tried different values ranging from 5 to 100 and selected k to be 30 as a good value. We then implemented the spread crossing calculations [3] by adding a new column to the data set. It has three possible values: "stat" for no spread crossing, "up" for an upward spread crossing and "down" for a downward spread crossing.

The next step was to implement a trading strategy based on the spread crossing direction:

At event i:

- If no spread ("stat") → do nothing
- If upward ("up") → buy at best ask at event i and sell at best bid at event i+30 (i.e. buy long sell short)
- If downward ("down") → buy at best ask at event i+30 and sell at best bid at event i (i.e. buy short sell long)

Once we implement this trading strategy, we first verify that it does what we intended it to do (i.e. making profit). Then we validate it by implementing our predictive models. We selected support vector machines (SVM) to fit our model for the reason explained in the following section. After selecting the kernels and tuning the kernel parameters for the SVM model as shown in section 4.2, we trained it and tested it. Then we calculated its accuracy which was around 99.17%. We then validated the model by applying it on the test data set. Our trading strategy led to a positive profit.

3.3 WHY SUPPORT VECTOR MACHINES (SVM)

Predicting the direction of mid-price movement and spread crossing (target variable) is multi classification problem with three categorical values ("up", "down" and "stat"). These target values have been calculated from various attributes from the order and the message book (as discussed in the previous section). In other words, the target values are known here. Therefore, we chose SVM as it is a supervised learning method that performs multi-class classification. After some training, SVM automatically carries out the classification of new data without user intervention.

Besides, the predictors and the target variable for our problem have collinearity and in some cases multicollinearity, meaning they are derived from one or more predictors. SVM is a good

choice for this problem as it is not sensitive to collinearity or multicollinearity. SVM also resists over fitting which helps getting better predictive performance with test data.

We also used Decision Tree (DT) to predict the direction of mid-price movement and spread crossing and compared the performance with SVM. As we expected, SVM out performed DT.

4. MODELING WITH R

4.1 DATA VISUALIZATION

The shape of the LOB is shown in figure 1 below. Also, the additional features added (i.e. mid-price and spread) to our model are shown in figure 1 and they are explained in the following section. A snapshot of the LOB is shown in figure 2 with the volumes of ask and bid at a specific point of time for the "orderbook" file of AMAZON Company. Figure 3 shows the relative depth of the level 10 data from the "orderbook" file of AMAZON too where level 1 shows the best ask and best bid and level 2 shows the second best ask and bid and so on. The levels following level 1 and their corresponding volumes can give an insight on the future behavior of the order book.

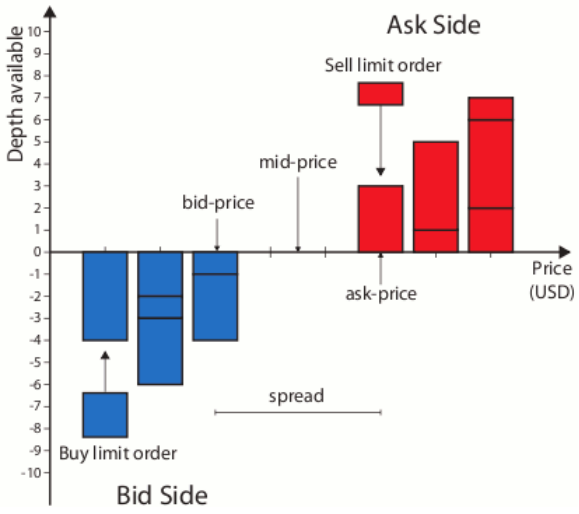

Figure 1: The shape of the LOB [4]

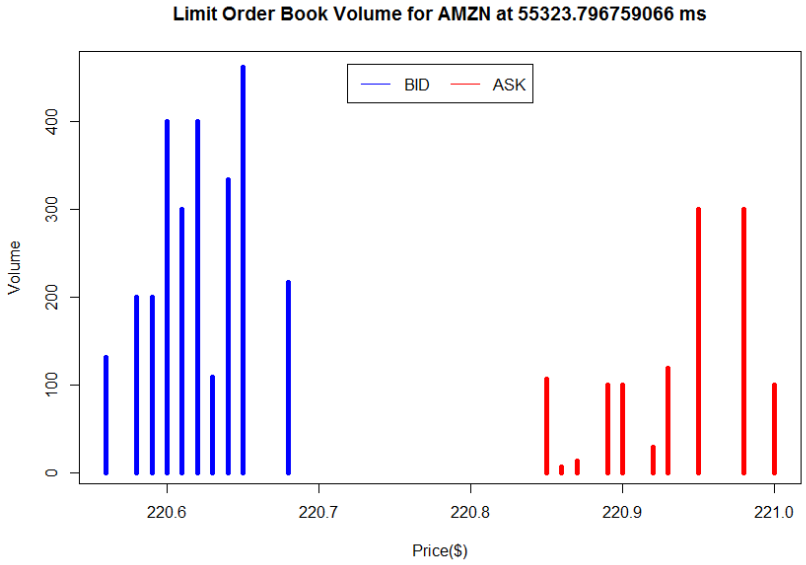

Figure 2: Snapshot of the LOB [5]

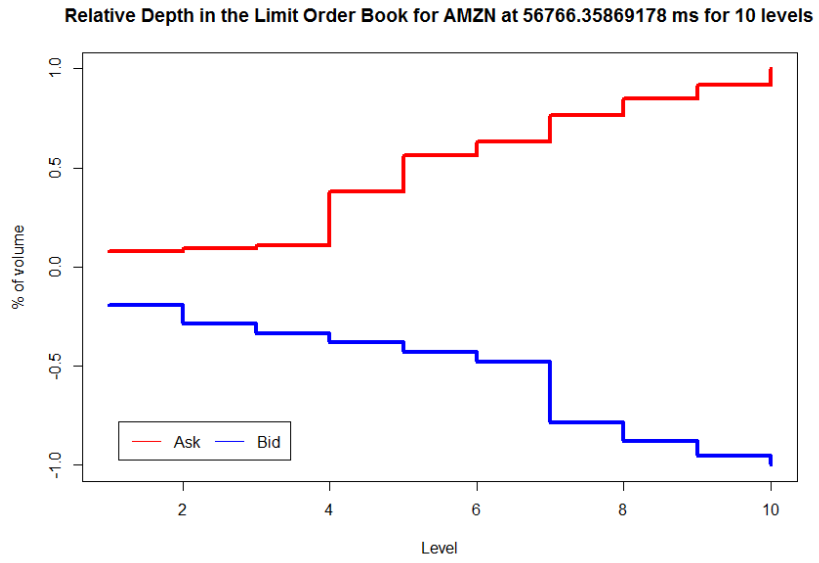

Figure 3: Relative depth in the LOB [5]

4.2 MODEL

For our model (R code is shown in Appendix B), we have used the level ten data that belongs to AMAZON that was obtained on June 21, 21 and available in [5]. We have first merged both

"orderbook" and "message" files into one file called "data". So, we originally have four available features for each level which are: "best ask price", "best ask volume", "best bid price", and "best bid volume" which add up to forty features since we have ten levels data. Moreover, we have added two additional features to each level [3] which are: "midprice" and "spread" and they can be calculated as per shown below:

Hence, we have a total of sixty features that will be considered in our model. The target of our model is the occurrences of the spread crossings, "spread_price", that can be determined as shown below based on the number of events k specified by the user:

$$\begin{cases} do \end{cases}$$

Where "stat" mean stationary (i.e. no price crossing). The model has been developed on a sample data of 60,000 records where they were split into two sets: train set (75% of the data) and test set (25% of the data). Two different classification models are built to predict the target. The first model is built using SVM with the parameters shown in table 2 below that are obtained after tuning the model. The second model is built using DT.

Table 2: SVM model parameters

Parameter	value
Type	C-svc C classification
Kernel	polydot Polynomial kernel
Kpar (kernel parameters)	list(degree=2)
C (cost of constraints violation)	0.25

A confusion matrix was obtained for each model as shown below in table 3 and 4 respectively in order to calculate the accuracy of their predictions and pick the best to be used for calculating the desired output of the project, *the profit*.

Table 3: Confusion matrix for the SVM model

SVM Model		*True* Classes		
		up	down	stat
Predicted Classes	up	0	1	1
	down	0	38	0
	stat	123	0	14837

Table 4: Confusion matrix for the DT model

DT Model		*True* Classes		
		up	down	stat
Predicted Classes	up	0	0	29
	down	11	16	3235
	stat	112	23	11574

As a result, the SVM model was more accurate as it has 99.17% prediction accuracy whereas the DT model has 77.27% prediction accuracy. Hence, the SVM model is chosen to for the next step. Finally, the profit is obtained using the strategy rule described in section 3.2.

4.3 DISCUSSION

After running our model in R considering the trading strategy in section 3.2, the profit was $5,999 from trading 100 shares when the spread crossing is “up” and 1000 shares when it is “down”. Several parameters can result in different values for the profit:

- The trading strategy considered by the user.
- The number of events k that is considered in fitting the model as well as calculating the profit.
- The number of shares to be traded when the target, spread crossing, is “up” or “down”.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The objective of this project was to study the limit order book for high frequency trading using historical data obtained from [5]. We used sixty predictors for 10 level data based on best ask and bid prices and volumes. Our main target was to predict the occurrence and direction of bid-ask spread crossing. We developed two predictive models (SVM and DT) to predict our target, i.e. spread_price considering *k* number of events. However, we focused on the SVM model since it had a higher accuracy.

We implemented a trading strategy based on the spread crossing direction and number of events which resulted in making profit. The profit amount can be affected by considering different trading strategies, thus changing the number of events and the number of shares will impact the total profit.

5.2 FUTURE WORK

In the future, we would like to try different predictive models and compare their performances with the suggested SVM model using criteria such as accuracy and computation time. Moreover, since we could not produce any profit consistently using the predictive models for mid-price movement, we would like to explore if adding mid-price movement as a predictor would increases the performance for the predictive model. We could also limit the number of features by selecting the most important ones. Techniques such as Information Gain could be used.

Finally, the effectiveness of the suggested model could be tested on low frequency trading, so that people can use this for personal gain/profit.

6. APPENDICES

APPENDIX.A

As explained in [5], LOBSTER generates a "message" and an "orderbook" file for each active trading day of a selected ticker. The "orderbook" file contains the evolution of the limit order book up to the requested number of levels. The "orderbook" file contains indicators for the type of event causing an update of the limit order book in the requested price range. All events are time stamped to seconds after midnight, with decimal precision of at least milliseconds and up to nanoseconds depending on the requested period.

"Time (sec)	Event Type	Order ID	Size	Price	Direction
⋮	⋮	⋮	⋮	⋮	⋮
34713.685155243	1	206833312	100	118600	-1
34714.133632201	3	206833312	100	118600	-1
⋮	⋮	⋮	⋮	⋮	⋮

Figure 4: snapshot from a "message" file [5]

Variable explanation for "message" file:

- Time: Seconds after midnight with decimal precision of at least milliseconds and up to nanoseconds depending on the period requested
- Event Type:
    - 1: Submission of a new limit order
    - 2: Cancellation (partial deletion of a limit order)
    - 3: Deletion (total deletion of a limit order)
    - 4: Execution of a visible limit order
    - 5: Execution of a hidden limit order
    - 7: Trading halt indicator (detailed information below)
- Order ID: Unique order reference number
- Size: Number of shares
- Price: Dollar price times 10000 (i.e. a stock price of $91.14 is given by 911400)
- Direction:

- -1: Sell limit order
- 1: Buy limit order
- Note: Execution of a sell (buy) limit order corresponds to a buyer (seller) initiated trade, i.e. buy (sell) trade.

Ask Price 1	Ask Size 1	Bid Price 1	Bid Size 1	Ask Price 2	Ask Size 2	Bid Price 2	Bid Size 2	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1186600	9484	118500	8800	118700	22700	118400	14930	...
1186600	9384	118500	8800	118700	22700	118400	14930	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 5: snapshot from an "orderbook" file [5]

Variable explanation for "orderbook" file:

- Ask Price 1: Level 1 ask price (best ask price)
- Ask Size 1: Level 1 ask volume (best ask volume)
- Bid Price 1: Level 1 bid price (best bid price)
- Bid Size 1: Level 1 bid volume (best bid volume)
- Ask Price 2: Level 2 ask price (second best ask price)
- Ask Size 2: Level 2 ask volume (second best ask volume)
- ...

"message" and "orderbook" files:

The "message" and "orderbook" files can be viewed as matrices of size (Nx6) and (Nx(4xNumLevel)), respectively, where N is the number of events in the requested price range and NumLevel is the number of levels requested.

The k-th row in the "message" file describes the limit order event causing the change in the limit order book from line k-1 to line k in the "orderbook" file.

Consider the "message" and "orderbook" figures above. The limit order deletion (event type 3) in the second line of the 'message' file removes 100 shares from the ask side at price 118600. The change in the "orderbook" file from line one to two corresponds to this removal of liquidity. The volume available at the best ask price of 118600 drops from 9484 to 9384 shares.

APPENDIX.B

R code developed for this project:

```
# -----------------
# DATA PREPARATION:
# -----------------

# Data: AMZN - June 21, 2012 - Level 10
dataOB <- read.csv("orderbook.csv")  #load order book data
dataM  <- read.csv("message.csv")    #load message data
data   <- cbind(dataM, dataOB)       #merge two data sets

nlevels <- (dim(dataOB)[2])/4  #compute the number of order book levels


# NAME THE COLUMNS:
# 1. first 6 columns from the message book:
col.name <- c("Time","Type","OrderID","Size","Price","TradeDirection")

# 2. next 40 columns for predictors from the order book considering 10 level
# data:
for(i in 1:nlevels)
{col.name <- c(col.name, paste("ASKp",i,sep=""), paste("ASKv",i,sep=""),
               paste("BIDp",i,sep=""), paste("BIDv",i,sep=""))}
colnames(data) <- col.name

# 3. next 20 columns created as predictors too:
for(i in 0:nlevels-1){
  for(j in 1:nrow(data)){
   data[j,47+(2*i)]<- (data[j,7+4*i]+data[j,9+4*i])/20000
  data[j,48+(2*i)] <- (data[j,7+4*i]-data[j,9+4*i])/10000
   }
  }

for(i in 1:nlevels)
 {
  col.name <- c(col.name, paste("midprice",i,sep=""),
paste("spread",i,sep=""))
 }
colnames( data) <- col.name

rm('dataOB','dataM') #cleanup
# ------------------------------------------------------------------------


# ----------------
# TARGET VARIABLE:
# ----------------
data <- read.csv("ImportantData.csv")
DataSample <- data[1:60000,-c(1,62)]

# Our target variable is the spread crossing, "spread_price": either "up",
# "down" or "stat"
k=30
```

```
for(i in 1:(nrow(DataSample)-k) )
{
  if(DataSample$ASKp1[i] < DataSample$BIDp1[i+k]) {
    DataSample$spread_price[i] <- "up"
    }
  else if(DataSample$BIDp1[i] > DataSample$ASKp1[i+k]){
      DataSample$spread_price[i] <- "down"
    }
  else {
      DataSample$spread_price[i] <- "stat"
  }
}

# calculate the number of the spread crossing, "spread_price":
nup   <- nrow(DataSample[DataSample$spread_price=="up",])
ndown <- nrow(DataSample[DataSample$spread_price=="down",])
nstat <- nrow(DataSample[DataSample$spread_price=="stat",])
nup     #274
ndown   #214
nstat   #59512

#set traget variable for spread crossing y
DataSample$y <- NA
DataSample[DataSample$spread_price=="up",]$y   <- 1
DataSample[DataSample$spread_price=="down",]$y <- 2
DataSample[DataSample$spread_price=="stat",]$y <- 3
# ------------------------------------------------------------------------


# ---------
# MODELING:
# ---------

# split the data using to create a TRAIN and TEST sets of data (25% for
# "test"; 75% for "train"):
train <- DataSample[1:45000,]      #the train set
test  <- DataSample[45001:60000,]  #the test set


# MODEL 1: Support Vector Machine (SVM)
# -------------------------------------
library(kernlab)
SVM.model <- ksvm(y ~ ., data = train, type = "C-svc", kernel = "polydot",
                  kpar = list(degree = 2), C = 0.25, prob.model = TRUE)
SVM.model
SVM.pred <- predict(SVM.model, test[,-62])


# MODEL 2: Decision Tree (DT)
# ---------------------------
library(rpart)
DT.model <- rpart(spread_price  ~ ., data = train[,-62])
DT.pred  <- predict(DT.model, test[,-61], type="class")
```

```
# MODELS COMPARISON:
# ------------------

# Confusion Matrix:
SVM.CM <- table(pred = SVM.pred, true = test[,62])
SVM.CM

DT.CM <- table(pred = DT.pred, true = test[,61])
DT.CM

# Model Accuracy:
SVM.Accuracy <- (38+14837)/15000
SVM.Accuracy   #99.17%

DT.Accuracy <- (16+11574)/15000
DT.Accuracy    #77.27%


# --------------------
# PROFIT CALCULATION:
# --------------------

test$pred <- SVM.pred

Profit=0

for(i in 1:(nrow(test)-k)){
  if(test$pred[i] == 1){

    Profit=Profit+100*(test$BIDp1[i+k]-test$ASKp1[i])
  }
  if(test$pred[i] == 2){

    Profit=Profit+10000*(test$BIDp1[i]-test$ASKp1[i+k])
  }
}

Profit/10000
```

REFERENCES

[1] Definition of high-frequency trading – HFT. *Finantial Times*, 2014, http://lexicon.ft.com/Term?term=high_frequency-trading-_-HFT

[2] Glossary of Stock Market Terms. *NASDAQ Stock Market*, 2014, http://www.nasdaq.com/investing/glossary/

[3] Kercheval, A. N., & Zhang, Y. (2013). Modeling high-frequency limit order book dynamics with support vector machines.

[4] Limit order books. Oxford Centre for Industrial and Applied Mathematics, 2014, https://www0.maths.ox.ac.uk/groups/ociam/research/finance

[5] Limit Order Book System – The Efficient Reconstructor. *LOBSTER*, 2014, https://lobster.wiwi.hu-berlin.de/

[6] Mertens, J. F. (2003). The limit-price mechanism. *Journal of Mathematical Economics*, 39(5), 433-528.

[7] Order book (trading). *Wikipedia*, 2014, http://en.wikipedia.org/wiki/Order_book_%28trading%29