TECHNISCHE UNIVERSITÄT WIEN

DISSERTATION

Stochastic modeling and statistical properties of the Limit Order Book

ausgeführt zum Zwecke der Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften unter der Leitung von

Univ.-Prof. Dr. Dipl.-Math. Thorsten Rheinländer

Institut für Stochastik und Wirtschaftsmathematik (E105)
Forschungsgruppe Finanz- und Versicherungsmathematik

eingereicht an der Technischen Universität Wien
Fakultät für Mathematik und Geoinformation

von

MSc Dragana Radojičić

Matrikelnummer: 11727238

Wien, im Juli 2020

(Unterschrift Verfasser) (Unterschrift Betreuer)

TECHNISCHE
UNIVERSITÄT
WIEN

DOCTORAL THESIS

Stochastic modeling and statistical properties of the Limit Order Book

Submitted for the degree of Doctor of Natural Sciences

by

Dragana Radojičić
TU Wien

Advisor:

Prof. Dr. Thorsten Rheinländer
TU Wien

Co-advisor:

Prof. Dr. Friedrich Hubalek
TU Wien

Referees:

Prof. Dr. Rama Cont
University of Oxford

Prof. Dr. Markus Fulmek
University of Vienna

Vienna, July 2020

Acknowledgment

I am today deeply thankful that I was given the chance of exchanging my views with many intellectual and generous personalities during the past years.

First and foremost, I would like to express my deep gratitude to my advisor Prof. Dr. Thorsten Rheinländer for the guidance in all phases of my work, for many insightful comments, his constant support, and many constructive suggestions. Furthermore, I would like sincerely to thank him for sharing his expertise and inspiring me to learn more.

My special thanks and profound gratitude are given to Prof. Dr. Friedrich Hubalek for his input, for many valuable discussions and engagement through the learning process during my doctorial studies.

I would also like to thank the members of my dissertation committee: Prof. Dr. Rama Cont and Prof. Dr. Markus Fulmek, who have willingly shared their precious time for refereeing my thesis, despite their extremely busy schedule.

I am greatly indebted to Prof. Dr. Rama Cont for providing me the opportunity to spend two wonderful, rewarding and enriching weeks at the University of Oxford within the Oxford Mathematical and Computational Finance Group.

Further, many thanks to M.Sc. Simeon Kredatus for many stimulating and fruitful discussions, and many interesting insights.

I would also like to acknowledge the academic and technical support of the Research Unit of Financial and Actuarial Mathematics at the TU Wien. I have had a great time to be a PhD student and Teaching Assistant here. I owe my deepest thanks to Sandra Trenovatz for an atmosphere of friendship, as well as for help with bureaucracy and an encouragement. Many thanks to Maike and Christiana for a lively and joyful working environment in the office, as well as for many discussions of mathematical and non-mathematical nature.

Finally, warmly thanks to my family for providing me with support, unconditional

love, and encouragement throughout my years of studies and through the process of doing research and writing this thesis.

Last but not the least, I would like to thank my friends for supporting me spiritually throughout writing this thesis. I will forever be grateful for your love.

Abstract

The aim of this thesis is to study the dynamics behavior and statistical properties of the Limit Order Book. This thesis consists of two main parts.

In the first part, a stochastic limit order book model in discrete time and space, driven by a simple symmetric random walk, is introduced. Special interest is given to an order avalanche in this model, where an avalanche is a series of order executions where the length of the periods with no trade event cannot exceed $\varepsilon > 0$. The development of the order book dynamics is described as follows. The new orders are placed with a fixed displacement $\mu \geq 1$ from the mid-price, and a trade event occurs whenever the mid-price hits the price level where there is some volume. Therefore, the dynamics of the limit order book model leads to two trading mechanisms, namely Type I trades and Type II trades. A Type I trade occurs after excursions to the next running maximum of the mid-price process, while a Type II trade occurs if the price drops by μ or more and then increases by μ again. Our focus is mainly on the distribution of order avalanches' length. The probability generating functions for different quantities, such as the time to the first trade assuming it is a Type I trade, the time to the first trade assuming it is a Type II trade, the time to the first Type II trade, the avalanche length, are derived.

The practical part of this thesis aims to analyze the informativeness of the features extracted from the limit order book to classify each market data vector into the one of the labels from the set $S = \{\text{buy}, \text{idle}\}$. The label *buy* indicates a buy order shall be issued, while *idle* indicates a trader should idle. The customized order book data reconstruction system is proposed in this thesis to extract features of interest from the market data. In addition, the technical indicators are concatenated into the data set that was previously aggregated and prepared into the desired form for this research. In order to describe the behavior present in the market, the concepts of supervised learning is utilized. Computational experiments are performed to evaluate if the performance of the model based on the

Gated Recurrent Unit (GRU) architecture is improved when we select features utilizing the newly proposed methods. Moreover, quantified results proving the significant performance improvement obtained by selecting features with respect to the proposed feature selection methods are provided in this thesis.

Zusammenfassung

Das Ziel dieser Doktorarbeit ist, das dynamische Verhalten und statistische Eigenschaften des Limit Order Buchs zu untersuchen. Die Arbeit besteht aus zwei Hauptteilen.

Im ersten Teil wird das stochastische Limit Order Buch Modell in diskreter Zeit und mit diskretem Zustandsraum eingeführt, das von einer einfachen symmetrischen Irrfahrt (eng. Random Walk) getrieben wird. Besonderes Interesse liegt auf eine Order Avalanche, wobei eine Avalanche eine Reihe von Order Ausführungen ist, bei denen die Länge der Perioden ohne Handelsereignis $\varepsilon > 0$ nicht überschreiten darf. Die Dynamik des Order Buchs entwickelt sich wie folgt. Die neuen Order treten mit einer festen Verschiebung $\mu \geq 1$ vom Mittelpreis (eng. Mid-Price) auf, und ein Handelsereignis tritt immer auf, wenn der Mittelpreis ein Preisniveau erreicht, das positives Volumen aufweist. Daher führt die Dynamik des Limit Order Buch Modells zu zwei Handelsmechanismen, nämlich Typ I-Handel und Typ II-Handel. Ein Typ I-Handel findet statt, wenn die Exkursionen das nächste laufende Maximum des Mittelpreisprozesses erreicht, während ein Typ II-Handel auftritt, wenn der Preis um μ oder mehr fällt und dann wieder um μ steigt. Unser Fokus liegt hauptsächlich auf der Verteilung der Länge des Avalanches. Die wahrscheinlichkeitserzeugende Funktion für die verschiedenen Größen, wie die Zeit bis zum ersten Handel unter der Voraussetzung, dass es ein Typ I-Handel ist, die Zeit bis zum ersten Handel unter der Annahme, dass es sich um einen Typ II-Handel handelt, oder die Avalanche Länge, werden abgeleitet.

Der praktische Teil dieser Arbeit hat als Ziel zu analysieren, wie viele Informationen über die Merkmale, die aus dem Limit Order Buch extrahiert werden können, um jedem Marktvektor eines der Labels aus der Menge $S = \{\text{buy}, \text{idle}\}$ zuzuweisen. Das Label „Buy“ zeigt an dass ein Kaufvertrag (eng. Buy order) abgeschlossen werden soll und „Idle“ bedeutet, dass ein Händler im Leerlauf sein sollte. Das kundenspezifische Rekonstruktionssystem für Auftragsbuchdaten wird in dieser Arbeit vorgeschlagen um interessante

Merkmale aus den Marktdaten zu extrahieren. Zusätzlich werden die technische Indikatoren in den zuvor aggregierten Datensatz eingebunden und aufbereitet. Um das vorhandene Verhalten auf dem Markt zu beschreiben, werden die Konzepte des überwachten Lernens (eng. supervised learning) angewendet. Computerexperimente basierend auf der Gated Reccurent Unit (GRU) Architektur werden durchgeführt, um zu bewerten, ob die Leistung des Modells verbessert wird, wenn Merkmale unter Verwendung der neu vorgeschlagenen Methoden ausgewählt werden. Darüber hinaus belegen quantifizierte Ergebnisse die signifikante Leistungsverbesserung durch Auswahl von Merkmalen in Bezug auf die vorgeschlagenen Merkmalsauswahlverfahren, die in dieser Arbeit vorgestellt werden.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 28. Juli 2020

Dragana Radojičić

Einverständniserklärung zur Plagiatsprüfung

Ich nehme zur Kenntnis, dass die vorgelegte Arbeit mit geeigneten und dem derzeitigen Stand der Technik entsprechenden Mitteln (Plagiat-Erkennungssoftware) elektronisch-technisch überprüft wird. Dies stellt einerseits sicher, dass bei der Erstellung der vorgelegten Arbeit die hohen Qualitätsvorgaben im Rahmen der ausgegebenen der an der TU Wien geltenden Regeln zur Sicherung guter wissenschaftlicher Praxis - "Code of Conduct" (Mitteilungsblatt 2007, 26. Stück, Nr. 257 idgF.) an der TU Wien eingehalten wurden. Zum anderen werden durch einen Abgleich mit anderen studentischen Abschlussarbeiten Verletzungen meines persönlichen Urheberrechts vermieden.

Wien, am 28. Juli 2020

Dragana Radojičić

Table of contents

1

Introduction

The mathematical modeling of the limit order book (LOB), as well as assessing the empirical properties of the limit order book data, is crucial for describing financial market behavior and for understanding order book dynamics. Therefore, the aim of this thesis is to study the limit order book dynamics and stock market behavior by employing stochastic modeling and machine learning techniques.

The first part of this thesis (Chapter 1) is devoted to introducing the research topic and the main object of this research, namely the limit order book.

In Chapter 2, which is based on [56], the stochastic limit order book model is studied. Precisely, the focus is on a limit order book model with discrete time and space, such that the new orders are placed at a fixed distance $\mu \geq 1$ above the mid-price, and a trade event occurs whenever the mid-price reaches the price level on which there is some volume. In this model, two possible execution mechanisms are considered, namely Type I trades and Type II trades. A typical Type I trade occurs whenever the mid-price is higher than the price on which the previous trade had occurred. A Type II trade occurs after the mid-price decreases by the value that is at least μ, and then increases by μ. The special focus is on studying the distribution of avalanches, where an avalanche is a series of order executions such that a time period without a trade event is not longer than an $\varepsilon > 0$. In other words, orders in the limit order book may accumulate on some levels and they get executed when the mid-price process reaches those values. This phenomenon is followed with a sudden decrease in the number of orders and can be described as an order avalanche.

In addition, at the end of Chapter 2, the stochastic model in which the mid-price converges in distribution to a subordinated Wiener process is introduced and briefly studied.

Chapter 3, which is an enhanced version of [100] and [99], studies the informativeness of features extracted from a limit order book data, to classify a market data vector into the label 'buy' or 'idle' by using the Gated Recurrent Unit (GRU) network. During the data transformation part features including technical indicators are extracted from the limit order data. More precisely, the technical indicators enhance the data set previously prepared into the desired form for this research. After the data reconstruction, the methodology used to select features, which are plugged in the GRU network model, is proposed. Moreover, the performance of the GRU model fed with randomly chosen features is compared with the performance of the same model fed with features selected by incorporation of the proposed methodology. Finally, Chapter 4 summarizes the results presented in this thesis, and possible extensions and future work are indicated.

1.1 Financial markets and electronic markets

The financial market is a type of marketplace for buying and selling financial instruments such as bonds, stocks, currencies, and derivatives. There are different types of financial markets such as the stock market, bond market, commodity market, forex market, derivatives market, etc. The explanation of the financial objects related to the financial market is presented in [15]. Different financial assets are traded at the stock exchange, however, in this thesis the shares are mainly considered.

At the end of the last century, technology has modified the traditional stock exchange systems, which were physically located, into the electronic stock exchanges, which use an online trading platform. This leads to new forms of trading, namely Algorithmic Trading and High-Frequency Trading (HFT), which utilize the communication technology speed. In recent years, data scientists have almost replaced the role of a trader in the sense that they are creating and statistically verifying trading strategies in order to obtain profit in the stock market. Algorithmic trading (or automated trading) is a form of trading that uses a computer program to determine and execute trading strategies. Note that these trading strategies employ mathematics and high-speed computers, and they are derived by taking into consideration variables such as timing, price, quantity, etc. High-Frequency Trading is a special method of algorithmic trading given that the portfolio holding period is very short. It uses sophisticated algorithms and technological tools to analyze multiple markets and rapidly trade securities in seconds or fractions of a second. Electronic markets

have improved the world financial markets in the sense that they reduce transaction costs, increase market liquidity, and provide information publicly for traders.

1.2 The Limit Order Book

The main purpose of the central object of this thesis, namely the Limit Order Book (LOB), is to record all unexecuted incoming and outgoing orders. The LOB keeps track of all transactions and available orders, so that the participants could possibly interact. An order book acquires available information on a specific market by keeping track of the market dynamics and trade events that are consequences of participants' actions. All observed data (such as price, quantity, timestamp, etc.) is stored in a file, called the limit order book (see [1]). The basic concept of both limit orders and market orders is to buy or sell a stock. However, the difference is in the mode of how they get executed. A limit order allows the trader to offer the price for purchase or sale at which a trader is willing to buy or sell a stock. More precisely, a trader places a limit order, and offers the price for it and waits until a suitable counterparty. On the other hand, with a market order, you immediately execute the trade at the best actual market price.

The LOB is situated on a discrete price grid, such that each available price level is represented by the grid's point. Note that the LOB depicts the two-sided prices, namely the bid side, which records the orders that need to be bought, and the ask side, which records the orders that need to be sold. The minimum distance between two price levels is called a *tick*. For each present price, the LOB records available volume corresponding to that price. In the stock market, the term volume refers to the number of shares available for a particular price. The limit order book at t containing N levels is defined by vectors $P_t^a = (P_{t,1}^a, P_{t,2}^a, ..., P_{t,N}^a)$ and $P_t^b = (P_{t,1}^b, P_{t,2}^b, ..., P_{t,N}^b,)$ representing the best N prices on the ask and bid side respectively, and by vectors $V_t^a = (V_{t,1}^a, V_{t,2}^a, ..., V_{t,N}^a)$ and $V_t^b = (V_{t,1}^b, V_{t,2}^b, ..., V_{t,N}^b,)$ describing the number of shares available for trade at t corresponding to the prices present in the vectors P_t^a and P_t^b respectively. The lowest price at the time t on the ask side $P_{t,1}^a$ is called the best ask price, while the highest price at the time t on the bid side $P_{t,1}^b$ is called the best bid price. See Figure 1.1 for an example of Apple Company stocks' limit order book snapshot.

In addition, to describe the LOB at time t, the two following quantities are derived from the best ask price $P_{t,1}^a$ and the best bid price $P_{t,1}^b$.

- *The mid-price*, an arithmetic average of the best bid and best ask price:

$$MidPrice_t = \frac{1}{2}(P^a_{t,1} + P^b_{t,1}).$$

- *The QuotedSpread*, defined as a difference between the best ask and the best bid price:

$$QuotedSpread_t = P^b_{t,1} - P^a_{t,1}.$$

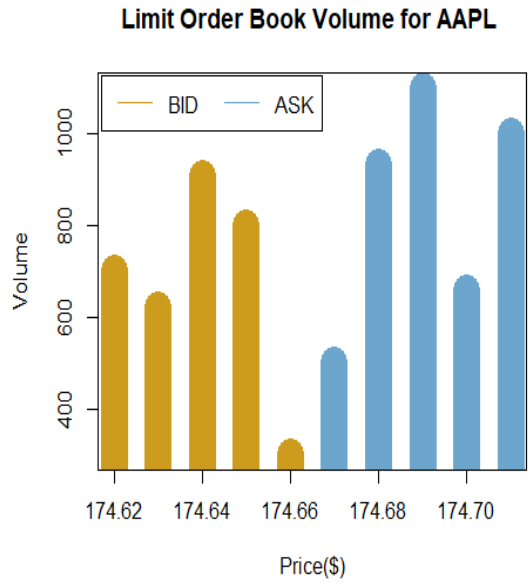

Figure 1.1: *Snapshot of the NASDAQ limit order book for AAPL shares (Jan 08, 2018) at 13:09:54 for 5 levels. Available buy/sell orders (depicted by gold/blue) are placed on the bid/ask side . The best bid price is* \$174.66*, while the best ask price is* \$174.67*.*

Note that an order placed at price $U > MidPrice$ is a sell order placed on the ask side, while an order placed at price $U < MidPrice$ is a buy order placed on the bid side. The comprehensive overview of the mathematical concept of the limit order book (LOB) is established in [61, Section II] and in [1].

1.3 Motivation

The significance of stock market studies have been confirmed by many research projects and publications on the topic, as well as by the interest of many financial institutions.

The last decades has brought a new way of trading, namely automated High Frequency Trading (HFT). This has reduced the Financial Transactions Tax (FTT), known also as Tobin tax, which although being very small is nevertheless non-zero. Thus, it leads to times of high distress of financial markets. This phenomenon is followed by occurrences of *flash crashes*, which occur when the security prices deeply fall down in a short time. A *flash crash* can occur due to several factors including high-frequency trading, see [18].

One of the best known events of this kind is the one that has occurred on 6th of May 2010 at 2:32 p.m. EDT, when stock indexes such as the Dow Jones Industrial Average, S&P 500 and Nasdaq Composite, incurred a large loss within minutes and recovered most of the losses within 36 minutes, see [73]. For example, during this flash crash, the Dow Jones had a loss of 9.5% and recovered within 15 minutes. Another example is the Singapore 2013 flash crash when some stocks lost up to 87 percent of their value.

In Section 2, the focus is on studying the distributional properties of this phenomenon in a basic but non-trivial model of the limit order book. Motivated by the self-organized criticality (SOC) property of dynamical systems, and inspired by the work presented in [111], the study of the *order avalanche length* is introduced. Therefore, the special focus of Section 2 is on the avalanche length object in the context of the limit order book.

Stock markets are producing a huge amount of empirical data. According to the earlier research (see [127], [32] and [92]) financial markets are informative and possess predictive properties. Moreover, the emergence of automated trading has caused that most trades are occurring via automated electronic markets (see [78]). The electronic stock exchanges have substituted the traditional physically located stock exchange. The electronic marketplace mechanism is comprehensively described in [22].

There is potential in employing Artificial Intelligence in order to develop algorithms that analyze historical data and to define quantitative trading strategies. Many researchers are interested in employing various machine learning approaches for financial forecasting, which is usually a very challenging task.

Section 3 studies the performance of a neural network model to predict if it is a good time to buy trade or if it is better to idle. Furthermore, the performance of the model when fed with different groups of features is compared.

2

On a binomial limit order book model

This chapter is mainly based on [56]. We introduce a non-trivial limit order book toy model where new orders get placed at a fixed distance $\mu \geq 1$ above the mid price. The orders get executed when the mid-price hits the price level on which they are placed. Our focus is mainly on the distribution of the order avalanche length, where an avalanche length for us is a series of order executions where the length of periods with no trade event cannot exceed $\varepsilon > 0$. Furthermore, the trade event occurs whenever the mid-price hits the price level on which there is some volume. Therefore, the introduced dynamics of the limit order book model leads to two trading mechanisms, namely Type I trades and Type II trades that will be described later.

Particularly, we pay special attention to how the intrinsic execution of the limit orders occurs. Note that the only possible execution mechanism is that mid-price reaches the price level at which the limit orders are placed, and any other cancellation mechanism is ignored. The limit order distribution is interpreted as a Dirac measure sitting on a level μ away from the mid-price, i.e. new orders are placed at a fixed displacement μ above the mid-price. This concept leads to an approachable, but nonetheless non-trivial execution mechanisms.

The structure of this Chapter is as follows. In Section 2.2, a discrete limit order book model driven by a simple symmetric random walk and a key quantity of our research, namely avalanche lengths are introduced. Then, the probability generating function and Brownian excursion limit of the simplified avalanche length are presented in Section 2.3. In addition, Section 2.4 contains limit results and the generating function for the full avalanche length. The generating function of the time to the first trade, if the order book

is initially empty, is presented in Section 2.5. The binomial limit order book model where the mid-price converges in distribution to a subordinated Wiener process is introduced in Section 2.6.

2.1 Related Literature

In recent years, there has been increasing interest in the study of dynamics of the limit order book, both from a theoretical and a practical point of view, see [1]. Modeling the dynamics of the order book as an interacting Markovian queueing system is exhibited in [31], and the analytical tractability of the model provides the relation between order flow and price dynamics. In [76] authors study a stochastic auction model in which buy and sell orders arrive following independent renewal processes, and the numbers of present orders are modeled by stochastic processes and furthermore their limiting distributions are investigated. Delattre et al. [41] investigate the efficient price that can be employed in practice, and also they develop statistical price estimation using the order flow. In [2] the authors investigate the order book model as a multidimensional continuous-time Markov chain, and furthermore using the functional central limit theorem the authors show that the rescaled price process converges to a Brownian motion. In [9] the authors investigate a stochastic LOB model that converges to a continuous-time limit, and the limits of the buy and sell volume densities are modeled by SPDEs (stochastic partial differential equations) coupled with a two-dimensional Brownian motion. A model in which limit order book dynamics depend on the available prices and volume corresponding to the current prices is introduced and studied in [66]. In particular, in [66] authors concluded that when the order size and tick size converges to zero, by a weak law of large numbers, the volume densities tend to non-linear PDEs, that are coupled with non-linear ODEs corresponding to the best ask and best bid price. Furthermore, in [67] the authors introduce a two-sided limit order book stochastic model and prove the limit theorem when the tick size converges to zero, so the best ask and best bid price processes are modeled by a two-dimensional reflected Brownian motion.

A simple order book model driven by arithmetic Brownian motion has been studied in [40], [102] and [110]. These studies lead to interesting known and new Brownian path functionals involving the theory of Brownian excursions. In the past it has been shown that such results can be illustrated, found, and often even proven by limit arguments with

or from corresponding results for random walks. For example, let us refer to [120], [112], [113], [35], [38], [37], [81], [34], [13] and in particular [109], [36] and [95].

2.2 The binomial limit order book model

2.2.1 The basic setting

Let $\{S_n : n \geq 0\}$ be a simple symmetric random walk defined on a probability space $(\Omega, \mathscr{F}, P)$. This random walk $\{S_n : n \geq 0\}$ we interpret as the *mid-price* process in our model. There exist various approaches for modeling mid-price in the limit order book models, for a general discussion on different *price substitutes* see [41]. Note that in the following we use the mixed index and function notation freely, i.e. S_k and $S(k)$ have the same meaning.

Definition 2.1. Let $\{V(n,u) : n \in \mathbb{N}_0, u \in \mathbb{Z}\}$ be a two-parameter order volume process, where $V(n,u)$ denotes the volume of orders at time n at price level u.

We are not interested in the size of the order volume, and therefore we distinguish only the following two cases: if there is an order at time n at price level u, indicated by $V(n,u) > 0$, or if there is no order, indicated by $V(n,u) = 0$.

Firstly, let us assume that the limit order book is initially full, i.e.

$$V(0,u) = I_{u \geq 0}, \quad u \in \mathbb{Z}. \tag{2.1}$$

Later, we assume that we start with an initially empty order book[1]. From now on the focus is only on the ask side of the order book, since by symmetry all calculations can be derived also for the bid side. Moreover, we ignore the order cancellations, although later the dynamics that also include order cancellations are introduced.

Let $\mu \geq 1$ be a fixed integer spread parameter that models the distance from the mid price at which orders get placed. The order book dynamics development is interpreted as follows. If there is some volume at the price level $u = S_n$ at time n, i.e. $V(n,S_n) > 0$, all orders sitting at that price level get executed. Therefore, we have $V(n+1,S_n) = 0$ at the next time step $n+1$. Note that $V(n+1,S_n) = 0$ also holds if there was no order at time

[1] Note that there are exchanges that clear the order book before a new trading day starts, for example the Istanbul Stock Exchange, see [117]

n at the price level $u = S_n$, i.e. if $V(n, S_n) = 0$. Furthermore, a new order will be placed at each step of the random walk at a distance μ above the price S_n, therefore we have $V(n+1, S_n + \mu) = V(n, S_n + \mu) + 1$. The dynamics that describe how the order book on the ask side evolves in time are summarized in the following equation

$$V(n,u) = \begin{cases} 0 & \text{if } u = S_{n-1} \\ V(n-1,u)+1 & \text{if } u = S_{n-1}+\mu \\ V(n-1,u) & \text{otherwise,} \end{cases} \qquad n \geq 1, u \in \mathbb{Z}. \tag{2.2}$$

Note that if we also consider the bid side the new order will be placed at a distance μ below the price S_n. Therefore, the dynamics that include both bid and ask side are summarized in the following equation

$$V(0,u) = 0, \quad V(n,u) = I_{\{S_{n-1} \neq u\}}[V(n-1,u) + I_{\{|S_{n-1}-u|=\mu\}}], \quad n \geq 1, u \in \mathbb{Z}. \tag{2.3}$$

Moreover, if order cancellations are also considered, they are interpreted as follows. We assume that no orders are placed at a distance grater or equal than $L > 0$ from the mid-price, and in case that an order has been submitted at price u before, that order is canceled as soon as $|S_n - u| \geq L$. In this case, the limit order book, including both the ask and bid side, evolves according to the following equation

$$V(n,u) = \begin{cases} 0 & \text{if } u = S_{n-1} \text{ or } |S_{n-1}-u| \geq L \\ V(n-1,u)+1 & \text{if } |S_{n-1}-u| = \mu \\ V(n-1,u) & \text{otherwise,} \end{cases} \qquad n \geq 1, u \in \mathbb{Z}. \tag{2.4}$$

Remark 2.1. *Note that all pictures that are included in this chapter are generated by a program that simulates the dynamics established in the equation (2.2). The program is written in the programming language C and it generates a latex file that, using the tikz package and its commands, generates the figure. The time is represented horizontally, price vertically, and the size of the order is ignored (the volume is either zero, and thus not depicted, or some positive number, which is represented with a square).*

In the following, the focus is on the ask side and order cancellations are ignored, thus we consider the dynamics established in the equation (2.2), which can be rewritten as

$$V(0,u) = 0, \quad V(n,u) = I_{\{S_{n-1} \neq u\}}[V(n-1,u) + I_{\{u=S_{n-1}+\mu\}}], \quad n \geq 1, u \in \mathbb{Z}. \tag{2.5}$$

Figure 5.1 (in Appendix) illustrates the dynamics development of the order book and the placement of new orders on the ask side, as summarized in equation (2.5). Note that those figures are generated by a program, written in the programming language *C++*, which simulates dynamics presented in equation (2.2).

2.2.2 Type I and Type II trades, trading times and intertrading times

A trade event occurs at time n if the mid-price reaches the price level U, i.e. $U = S_n$, and there is some volume at the price level U, i.e. if $V(n, S_n) > 0$.

Definition 2.2. The trading times $\{\tau_i : i \geq 0\}$ and the intertrading times $(T_i)_{i\geq 1}$ are defined by

$$\tau_0 = 0, \quad \tau_i = \inf\{n > \tau_{i-1} : V(n, S_n) > 0\}, \quad T_i = \tau_i - \tau_{i-1}, \quad i \geq 1. \tag{2.6}$$

This is a new approach to order book modeling, in which the new orders get placed with a fixed displacement from the mid-price and get executed whenever the mid-price reaches the price level on which there is some volume.

This dynamics setup leads us to consider two execution mechanisms, namely Type I trades and Type II trades.

- If $S(\tau_i) > S(\tau_{i-1})$ for $i \geq 1$ the i-th trade is called a Type I trade.
- Otherwise, the i-th trade is called a Type II trade.

A Type II trade occurs if the price drops by μ or more, and then increases by μ or more again. Note that the trade that occurs at the initial time ($n = 0$) at price level $u = 0$, is defined to be a Type II trade. If a Type II occurs within a very short time interval of length less than $\varepsilon > 0$, it is called a *flash-crash* trade. Intuitively, Type II trades, and in particular, flash-crash trades are less frequent than Type I trades. A Type II trade is unlikely to occur for small values of ε and for large values of μ.

Figure 2.1 displays an example of a typical Type I trade path (a blue rhombus depicts a Type I trade), while Figure 2.2 displays an example of a typical Type II trade path (a blue rhombus depicts a Type I trade, while a red rectangle depicts a Type II trade). The particular code that generates Figure 2.2 is presented in Appendix 5.1.

The following lemma establishes a sufficient condition for a Type I trade to occur, and it will motivate us to introduce *simplified trading times* and *simplified intertrading times* later.

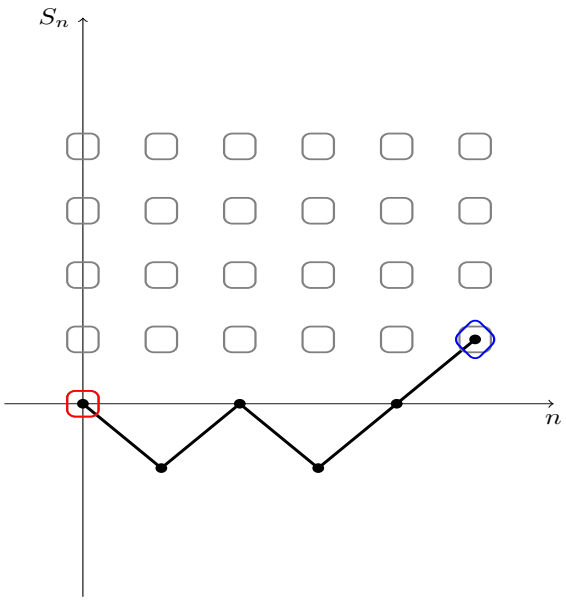

Figure 2.1: *Example of a Type I trade path. Blue rhombuses represent Type I trades, and red squares represent Type II trades.*

Lemma 2.1. *If $n \geq 1$ is a strict ascending ladder time of the random walk, then a Type I trade takes place at time n.*

Proof. Let $n \geq 1$ be a strict ascending ladder time. Then, $S_n > \max(S_0, \ldots, S_{n-1}) \geq 0$ and there was no trade at level S_n or higher before n. The order book is initially full (see (2.1)), so $V(0,u) > 0$ for $u \geq 0$. Therefore, it follows $V(n,S_n) > 0$, and thus there is a trade at time n, and at a price level S_n which is higher than the level of the last trade. □

Corollary 2.1. *If $S_n > \max(S_{\tau_i}, \ldots, S_{n-1}) \geq 0$, and a Type I or a Type II trade trade occurred at time $\tau_i < n$, then a Type I trade takes place at time n.*

Intuitively, a trade on the ask-side occurs when the mid-price moves up. Motivated by Lemma 2.1, and also in agreement with [40], with the following definition we introduce *simplified trading times* and *simplified intertrading times*.

Definition 2.3. Simplified trading times $\{\rho_i : i \geq 0\}$ and simplified intertrading times $(R_i)_{i\geq 1}$ are defined as

$$\rho_0 = 0, \quad \rho_i = \inf\{n > \rho_{i-1} : S_n > S(\rho_{i-1})\}, \quad R_i = \rho_i - \rho_{i-1}, \quad i \geq 1. \tag{2.7}$$

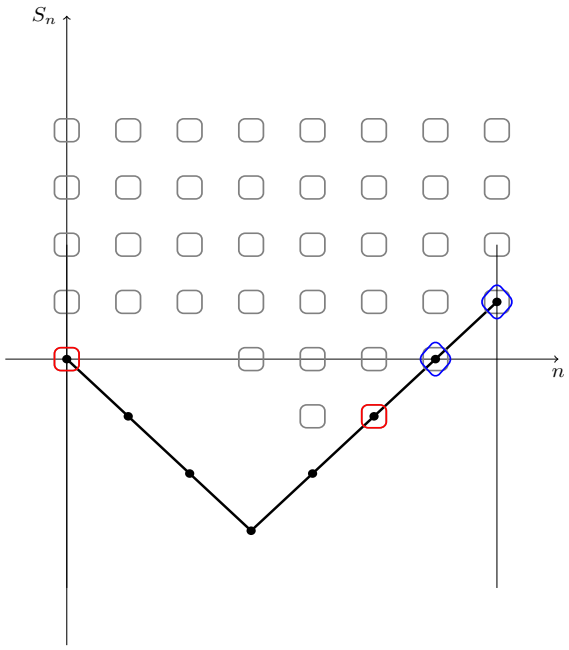

Figure 2.2: *Example of a Type II trade path. Blue rhombuses represent Type I trades, and red squares represent Type II trades. Illustration with* $\mu = 2$. *A Type II trade is followed by two consecutive Type I trades.*

Remark 2.2. *Note that* $\{\rho_j : j \geq 0\} \subset \{\tau_i : i \geq 0\}$ *and that with Definition 2.3 we ignore Type II trades, but we may also ignore some Type I trades that occur after a Type II trade. Figure 2.3 illustrates a Type II trade which is followed by two consecutive Type I trades, of which first one is ignored by Definition 2.3.*

2.2.3 On the best ask price

Since the focus is on the ask side, a trade on the ask-side intuitively occurs when the mid-price moves up.

Let the best ask price processes $\{\alpha_n\}_{n\geq 0}$ be defined as follows

$$\alpha_0 = 0, \quad \alpha_{n+1} = \alpha_n + I_{\{\alpha_n = S_n\}} - I_{\{\alpha_n = S_n + \mu + 1\}}, \quad n \geq 0. \tag{2.8}$$

Note that the equation (2.8) describes the best ask price process by the mid-price process without referring to the order volume process. The equation (2.8) can be interpreted as

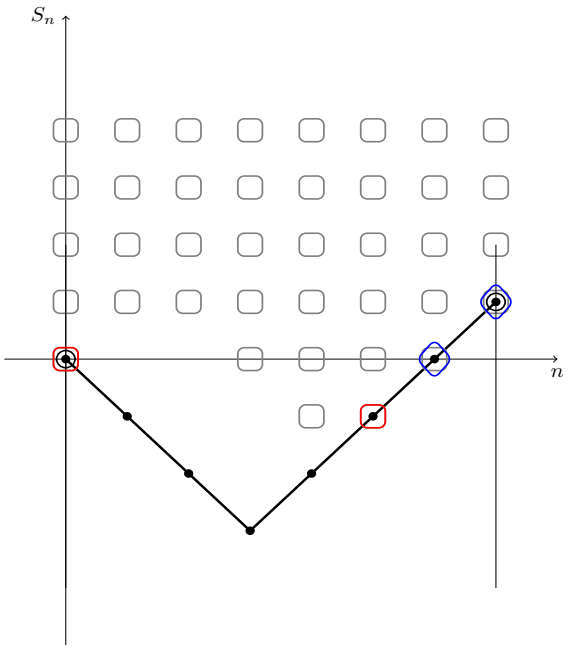

Figure 2.3: *Black circles depict trades that occur at the simplified trading times, blue rhombuses represent Type I trades, red squares represent Type II trades*

follows. If the mid-price hits the best ask price, i.e. $\alpha_n = S_n$, then the next best ask price is higher than the previous one. On the other hand, if $\alpha_n = S_n + \mu + 1$, i.e. the order is placed below the previous best ask price, then the next best ask price is lower than the previous one.

Lemma 2.2. *The following inequalities hold*

$$S_n \leq \alpha_n \leq S_n + \mu + 1, \quad n \geq 0. \tag{2.9}$$

Proof. Let us distinguish the following six cases: $\alpha_n = S_n$ and $S_{n+1} = S_n + 1$, $\alpha_n = S_n$ and $S_{n+1} = S_n - 1$, $\alpha_n = S_n + \mu + 1$ and $S_{n+1} = S_n + 1$, $\alpha_n = S_n + \mu + 1$ and $S_{n+1} = S_n - 1$, $S_n < \alpha_n < S_n + \mu + 1$ and $S_{n+1} = S_n + 1$, $S_n < \alpha_n < S_n + \mu + 1$ and $S_{n+1} = S_n - 1$. Then, this lemma is proved by induction on n for each case and for the step $n \mapsto n+1$.

Case 1: suppose that $\alpha_n = S_n$ and $S_{n+1} = S_n + 1$ holds. Then, $\alpha_{n+1} = \alpha_n + 1$ by (2.8). Thus, the following holds $S_{n+1} = S_n + 1 = \alpha_n + 1 = \alpha_{n+1} = S_n + 1 \leq S_{n+1} + \mu + 1$.

Case 2: suppose that $\alpha_n = S_n$ and $S_{n+1} = S_n - 1$ holds. Then, $\alpha_{n+1} = \alpha_n + 1$ by (2.8),

and $S_{n+1} = \alpha_n - 1 < \alpha_{n+1}$. Furthermore, since $\mu \geq 1$ we have

$$\alpha_{n+1} = \alpha_n + 1 = S_n + 1 \leq S_n - 1 + \mu + 1 = S_{n+1} + \mu + 1.$$

Case 3: suppose that $\alpha_n = S_n + \mu + 1$ and $S_{n+1} = S_n + 1$ holds. Then, $\alpha_{n+1} = \alpha_n - 1$ by (2.8). Therefore, $\alpha_{n+1} = \alpha_n - 1 = S_n + \mu < S_n + 1 + \mu + 1 = S_{n+1} + \mu + 1$. Further, since $\mu \geq 1$ the following holds $S_{n+1} = S_n + 1 = \alpha_n - \mu \leq \alpha_n - 1 = \alpha_{n+1}$.

The remaining three cases are proven in the same fashion by induction on n and for the step $n \mapsto n+1$. Furthermore, the program simulation for $\mu = 3$ has confirmed this lemma, see Figure 2.4. □

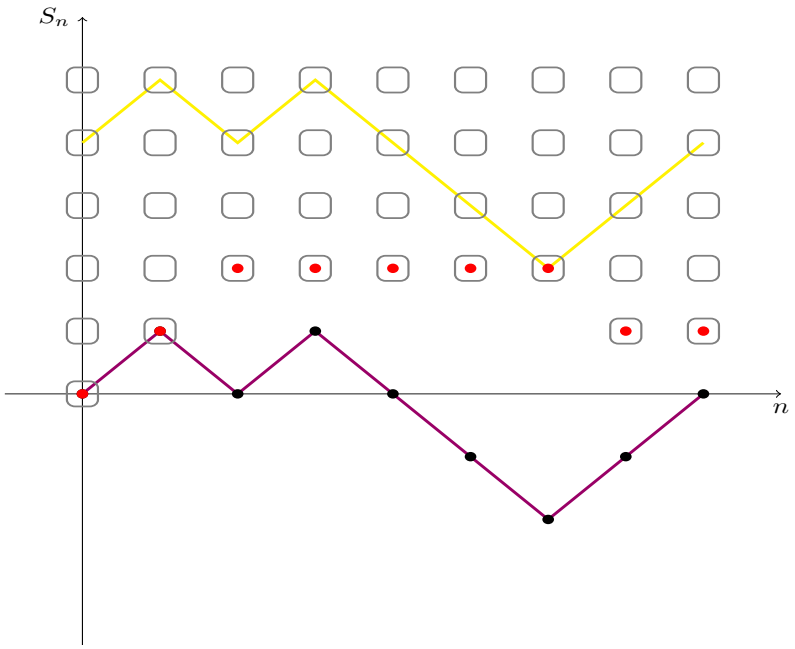

Figure 2.4: *The below line represents the mid-price process, the above yellow line represents* $S_n + \mu + 1$*, and the red points depict the best ask process,* $\mu = 3$.

Lemma 2.3. *For* $n \geq 0$ *we have that* $V(n,u) > 0$ *if and only if* $u \geq \alpha_n$.

Proof. This lemma is proved by induction on n. Since the limit order book is initially full (2.1), the claim is true for $n = 0$. Three different cases are distinguished to prove this lemma, and for each case induction on n for the step $n \mapsto n+1$ is used.

Case 1: Suppose that $S_n = \alpha_n$, then $\alpha_{n+1} = \alpha_n + 1$ by (2.8). Then, by the induction hypothesis $V(n,u) > 0$ if and only if $u \geq \alpha_n$. There is a trade at time n and therefore we have $V(n+1,S_n) = V(n+1,\alpha_n) = 0$. Furthermore, a new order is placed at distance μ above the mid-price, so $V(n+1,S_n+\mu) = V(n,S_n+\mu)+1 > 0$, and volume at all other price levels stays unchanged. Therefore, $V(n+1,u) > 0$ if and only if $u \geq \alpha_{n+1}$.

Case 2: Suppose that $\alpha_n = S_n + \mu + 1$. So $V(n,S_n) = 0$ since $S_n < \alpha_n$, and thus no trade takes place. A new order is placed at price level $S_n + \mu = \alpha_n - 1 = \alpha_{n+1}$, and therefore $V(n+1,\alpha_{n+1}) > 0$. All other positions are unchanged.

Case 3: Suppose that $S_n < \alpha_n < S_n + \mu + 1$. Then $\alpha_{n+1} = \alpha_n$. No trade takes place also in this case. The new order is placed at the price level $S_n + \mu \geq \alpha_n$. All other positions are unchanged. □

Corollary 2.2. *For $i \geq 1$ the following equality holds*

$$\tau_i = \inf\{n > \tau_{i-1} : S_n = \alpha_n\}. \tag{2.10}$$

Remark 2.3. *The trade happens exactly whenever the mid-price hits the best ask price, i.e. when $S_n = \alpha_n$, $n \geq 1$.*

Remark 2.4. *Figure 2.5 is generated by a program written in the programming language* C++*, which simulates dynamics presented in equation (2.2) and (2.8) , and it confirms the aforementioned results for the best ask price process $\{\alpha_n\}_{n\geq 0}$. Note that in Figure 2.5 red points depict the best ask price process $\{\alpha_n\}_{n\geq 0}$.*

2.2.4 The trading excursion process

Let us define the *trading excursion process* as a process that takes values in the set of simple random paths of finite length. Precisely, for $n \geq 0$ let

$$\mathbf{U}^{(n)} = \left\{(s_0,\ldots,s_n) \in \mathbb{Z}^{n+1} : s_0 = 0,\ |s_j - s_{j-1}| = 1 \text{ for } j = 1,\ldots,n\right\}, \tag{2.11}$$

and $\mathbf{U} = \bigcup_{n\geq 0} \mathbf{U}^{(n)}$. Furthermore, define $\mathscr{U}$ as the power set of U and define a discrete measure ν as $\nu(\{(s_0,\ldots,s_n)\}) = 2^{-n}$. The triple $(U,\mathscr{U},\nu)$ is a σ-finite measure space.

Definition 2.4. The *trading excursion process* $(e_i)_{i\geq 1}$ that takes values in U is defined by

$$e_{in} = S(\tau_{i-1}+n) - S(\tau_{i-1}), \quad i \geq 1, \quad 0 \leq n \leq T_i. \tag{2.12}$$

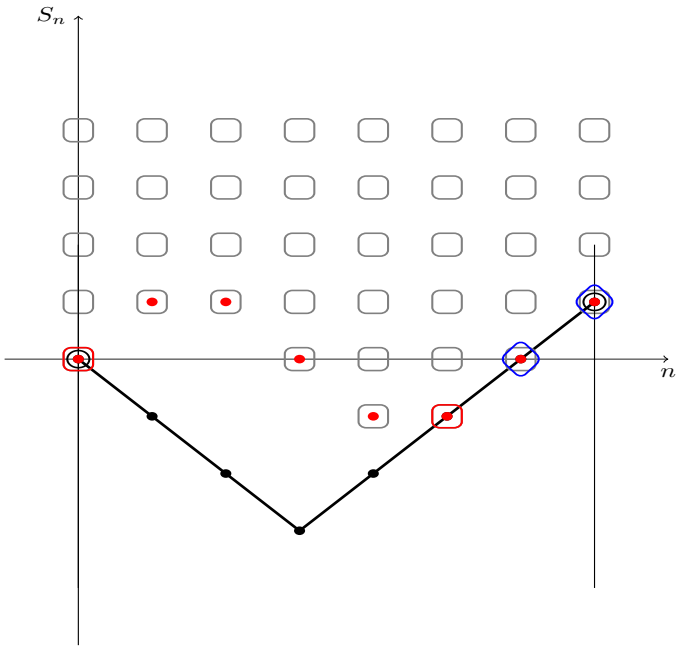

Figure 2.5: *Red points represent the best ask process, blue rhombus represent Type I trade, and red squares represent Type II trade.*

Remark 2.5. *For each $i \geq 1$ if $e_{iT_i} > 0$ a Type I trade occurs at time τ_i, otherwise if $e_{iT_i} \leq 0$ a Type II trade occurs at time τ_i.*

Lemma 2.4. *The trading excursions $\{e_i\}_{i\geq 1}$ are independent and identically distributed.*

Proof. Because of the Markov property of the simple symmetric random walk, the claim follows since $\tau_i, i \geq 0$ are stopping times for the random walk. To give some details, fix $i \geq 1$ and consider the processes

$$S'(n) = S(\tau_{i-1}+n) - S(\tau_{i-1}), \quad \alpha'(n) = \alpha(\tau_{i-1}+n) - \alpha(\tau_{i-1}), \quad n \geq 0. \tag{2.13}$$

□

Remark 2.6. *The trading excursion process is uniquely determined by the price process, but also the price process can be reconstructed from the trading excursion process by gluing the excursions together, following the same procedure as in the classical excursion theory, see [101, Prop.XII.2.5, P.482].*

2.2.5 The avalanche length

Motivated by the self-organized criticality (SOC) property of dynamical systems, and inspired by [102], our focus is on a key scalar quantity, namely the *avalanche length* in the context of financial markets.

Therefore, instead of studying the dynamics of the two-parameter volume process $\{V(n,u) : n \in \mathbb{N}_0, u \in \mathbb{Z}\}$, the aim is to study the avalanche length distribution. When there is a longer up-movement of the asset price, then a substantial portion of the order book get executed until the price drops again. This event can be interpreted as an order avalanche.

Note that the long up movements are rare for a simple symmetric random walk. If we consider the corresponding model in continuous time, the probability that Brownian motion increases on an interval of positive length is zero. Thus, an avalanche is considered as a period of trade executions such that times between consecutive trades is $\varepsilon > 0$ at most. The avalanche terminates when there is no trade for a period of length larger than ε.

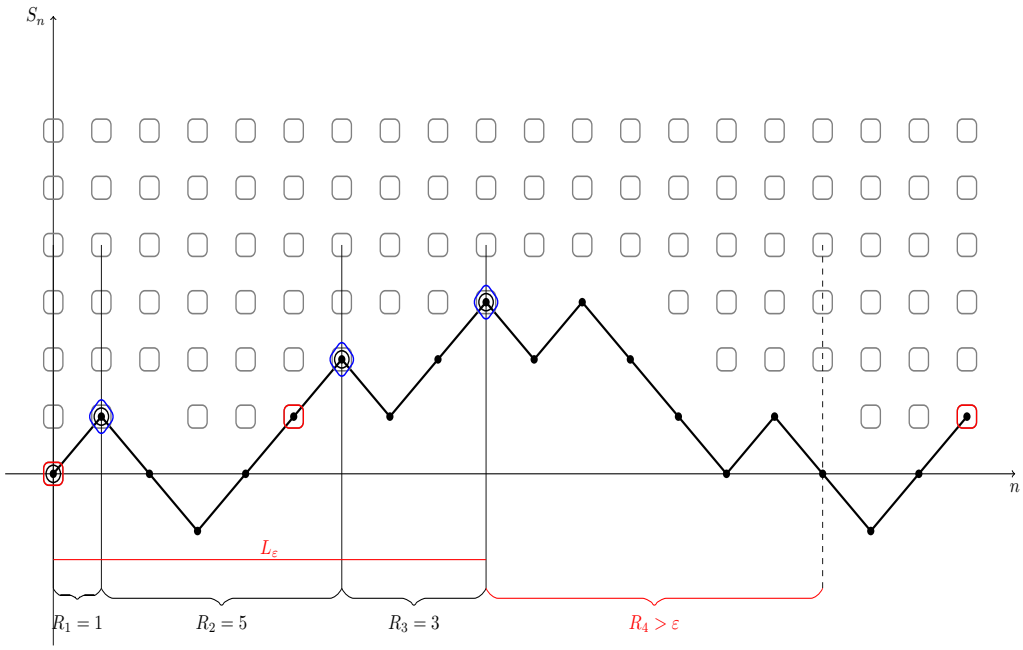

Figure 2.6: *Example of an avalanche without flash-crash trades when* $\varepsilon = 6$

Figure 2.6 illustrates an example of an avalanche length without flash-crash trades. For the simplified avalanche length we consider only Type I trades, and completely ignore (do

not take into account) Type II trades and even some Type I trades that follow Type II trade (as depicted in Figure 2.3).

Note that an avalanche may continue, even if the price drops, and recovers quickly in a short time interval of length less than ε, i.e. when a flash-crash occurs. This situation is illustrated in Figure 2.7 for $\varepsilon = 8, \mu = 2$. Formal definitions are given below.

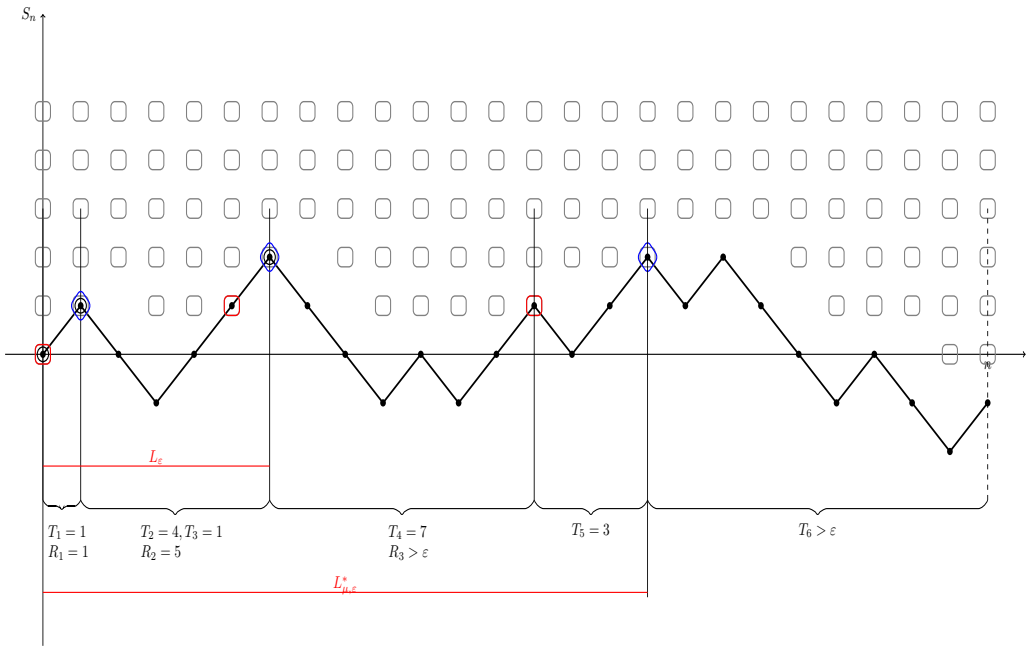

Figure 2.7: *The full avalanche length $L^*_{\mu,\varepsilon}$ versus simplified avalanche length L_ε, example with a flash-crash trade, $\varepsilon = 8, \mu = 2$.*

2.3 The simplified avalanche length

2.3.1 The probability generating function

The simplified avalanche length L_ε is defined as follows: if $k \geq 1$ and $R_1 \leq \varepsilon, \ldots, R_k \leq \varepsilon, R_{k+1} > \varepsilon$, then

$$L_\varepsilon = R_1 + \ldots + R_k. \tag{2.14}$$

An illustration of the simplified avalanche length example for $k = 3$ and $\varepsilon = 7$ is presented in Figure 2.6.

Proposition 2.1. *The probability generating function of the simplified avalanche length is given by*

$$E[z^{L_\varepsilon}] = \frac{P[R_1 > \varepsilon]}{E[1 - z^{R_1}; R_1 \leq \varepsilon] + P[R_1 > \varepsilon]}. \tag{2.15}$$

Proof. From (2.14) the following holds

$$\begin{aligned} E[z^{L_\varepsilon}] &= \sum_{k\geq 0} E[z^{R_1+\cdots+R_k} : R_1 \leq \varepsilon, R_2 \leq \varepsilon, \ldots, R_k \leq \varepsilon, R_{k+1} > \varepsilon] \\ &= \sum_{k\geq 0} E[z^{R_1} z^{R_2} \cdots z^{R_k} : R_1 \leq \varepsilon, R_2 \leq \varepsilon, \ldots, R_k \leq \varepsilon, R_{k+1} > \varepsilon] \end{aligned} \tag{2.16}$$

Since $\{\rho_j\}_{j\geq 0}$ are strict ascending ladder times, by following [51, XIII.1d, P.305] it is clear that $\{R_j\}_{j\geq 0}$ are independent and identically distributed. Therefore, we obtain

$$\begin{aligned} E[z^{L_\varepsilon}] &= \sum_{k\geq 0} E[z^{R_1} : R_1 \leq \varepsilon]^k E[1 : R_{k+1} > \varepsilon] = \frac{P[R_{k+1} > \varepsilon]}{1 - E[z^{R_1} : R_1 \leq \varepsilon]} \\ &= \frac{P[R_{k+1} > \varepsilon]}{E[1 : R_1 \leq \varepsilon] + E[1 : R_1 > \varepsilon] - E[z^{R_1} : R_1 \leq \varepsilon]} \\ &= \frac{P[R_{k+1} > \varepsilon]}{E[1 - z^{R_1} : R_1 \leq \varepsilon] + P[R_1 > \varepsilon]} = \frac{P[R_1 > \varepsilon]}{E[1 - z^{R_1} : R_1 \leq \varepsilon] + P[R_1 > \varepsilon]}. \end{aligned} \tag{2.17}$$

□

Remark 2.7. *The random variable R_1 represents the first passage time of the random walk through 1. Its distribution can be computed by the reflection principle, and can be found in [51, Theorem 2 of III.7, P.89]. Let $\varphi_{r,n}$ be the probability that the first passage through $r \in \mathbb{N}$ occurs at $n \in \mathbb{N}$. Then, by following [51, Theorem 2 of III.7, P.89] the probability $\varphi_{r,n}$ can be expressed as*

$$\varphi_{r,n} = \frac{r}{n} \binom{n}{\frac{n+r}{2}} 2^{-n}. \tag{2.18}$$

Therefore, the probability that the first passage of the random walk through 1 occurs at time $2n+1$, $n \geq 0$, is given by

$$P[R_1 = 2n+1] = \varphi_{1,2n+1} = \frac{1}{2n+1} \binom{2n+1}{n+1} 2^{-(2n+1)}.$$

The corresponding probability generating function (see [51, XIII,(4.10), P.315]) is

$$E[z^{R_1}] = \frac{1-\sqrt{1-z^2}}{z}. \tag{2.19}$$

Corollary 2.3. *Let a_k be the number of the first simplified Type I trade paths of length $k \geq 1$. Then, by employing the ability of the generating function to solve counting problems in order to choose distinct items from a set, and by expanding (2.19) with the binomial theorem (see Appendix 5.3), a_k can be expressed as*

$$a_k = \frac{1}{k}\binom{k}{(k+1)/2} \tag{2.20}$$

when k is odd, and $a_k = 0$ otherwise.

The code written in programming language *R* that contains three methods computing the number of paths corresponding to the simplified trades, is presented in Appendix 5.3.2, and results are summarized in Table 5.1.

The probability generating function in (2.15) can be rewritten in various different and more explicit ways.

For even ε, i.e. $\varepsilon = 2l$ the following holds

$$P[R_1 > \varepsilon] = \sum_{j=l}^{\infty} P[R_1 = 2j+1] = \frac{1+l}{1+2l}\binom{1+2l}{1+l}/2^{2l}. \tag{2.21}$$

For odd ε, i.e. $\varepsilon = 2l+1$ the following holds

$$P[R_1 > \varepsilon] = \sum_{j=l+1}^{\infty} P[R_1 = 2j+1] = \frac{2+l}{3+2l}\binom{3+2l}{2+l}/2^{2(l+1)}. \tag{2.22}$$

To summarize

$$P[R_1 > \varepsilon] = \frac{3+\varepsilon'}{2+\varepsilon'}\binom{2+\varepsilon'}{\frac{3+\varepsilon'}{2}}/2^{2+\varepsilon'}, \tag{2.23}$$

where ε' is defined to simplify notation and distinguish odd and even ε as

$$\varepsilon' = 2\left\lfloor \frac{\varepsilon-1}{2} \right\rfloor + 1.$$

Since

$$E[z^{L_\varepsilon}] = \frac{P[R_{k+1} > \varepsilon]}{1 - E[z^{R_1} : R_1 \leq \varepsilon]}, \tag{2.24}$$

the probability generating function of the simplified avalanche length can be written as

$$E[z^{L_\varepsilon}] = \frac{1 - \Phi_\varepsilon(1)}{1 - \Phi_\varepsilon(z)}, \tag{2.25}$$

where $\Phi_\varepsilon(z)$ is a polynomial, namely

$$\Phi_\varepsilon(z) = \sum_{k=0}^{(\varepsilon'-1)/2} \varphi_{1,2k+1} z^{2k+1}. \tag{2.26}$$

Note that the function $\Phi_\varepsilon(z)$ can be expressed in terms of the Gaussian hypergeometric function ${}_2F_1(a,b,c,z)$,

$$\Phi_\varepsilon(z) = \frac{1-\sqrt{1-z^2}}{z} - \binom{2+\varepsilon'}{\frac{3+\varepsilon'}{2}} \frac{z^{2+\varepsilon'}}{(2+\varepsilon')2^{2+\varepsilon'}} {}_2F_1(1,1+\frac{\varepsilon'}{2};\frac{5+\varepsilon'}{2};z^2). \tag{2.27}$$

Remark 2.8. *The hypergeometric function ${}_2F_1(a,b;c;z)$ is defined for $|z|<1$ by the power series*

$${}_2F_1(a,b;c;z) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n} \frac{z^n}{n!}. \tag{2.28}$$

See [50] for a systematic overview of identities involving hypergeometric functions.

Remark 2.9. *The hypergeometric function with parameter 1 can be expressed also in terms of (generalizations) of the associated Legendre Polynomials, see [3, 15.4.13, P.562].*

Corollary 2.4.

$$E[L_\varepsilon] = 2 + \varepsilon' - \frac{2+\varepsilon'}{3+\varepsilon'} \cdot 2^{2+\varepsilon'} \Big/ \binom{2+\varepsilon'}{\frac{3+\varepsilon'}{2}}, \tag{2.29}$$

$$\begin{aligned} \mathrm{Var}[L_\varepsilon] = {} & \frac{4}{3} \cdot (2 + 3\varepsilon' + \varepsilon'^2) - \frac{6+7\varepsilon'+2\varepsilon'^2}{3+\varepsilon'} \cdot 2^{2+\varepsilon'} \Big/ \binom{2+\varepsilon'}{3+\varepsilon'} \\ & + \frac{(2+\varepsilon')^2}{(3+\varepsilon')^2} \cdot 2^{4+2\varepsilon'} \Big/ \binom{2+\varepsilon'}{\frac{3+\varepsilon'}{2}}^2. \end{aligned} \tag{2.30}$$

Proof. Let $G_{L_\varepsilon}(z)$ be the probability generating function of the simplified avalanche length.

Then, by equation (2.25) and equation (2.26) we have

$$G_{L_\varepsilon}(z) = E[z^{L_\varepsilon}] = \frac{1-\Phi_\varepsilon(1)}{1-\Phi_\varepsilon(z)} = \frac{1-\sum_{n=0}^{(\varepsilon'-1)/2} \frac{1}{2n+1}\binom{2n+1}{n+1}2^{-(2n+1)}}{1-\sum_{n=0}^{(\varepsilon'-1)/2} \frac{1}{2n+1}\binom{2n+1}{n+1}2^{-(2n+1)}z^{2n+1}}, \tag{2.31}$$

and thus

$$G'_{L_\varepsilon}(z) = \frac{1-\Phi_\varepsilon(1)}{(1-\Phi_\varepsilon(z))^2} \cdot \sum_{n=0}^{(\varepsilon'-1)/2} \binom{2n+1}{n+1} 2^{-(2n+1)} z^{2n}. \tag{2.32}$$

Therefore, the following equalities hold

$$\begin{aligned} G'_{L_\varepsilon}(1^-) &= \frac{1}{1-\Phi_\varepsilon(1)} \sum_{n=0}^{(\varepsilon'-1)/2} \binom{2n+1}{n+1} 2^{-(2n+1)} \\ &= \frac{1}{1-\sum_{n=0}^{(\varepsilon'-1)/2} \frac{1}{2n+1}\binom{2n+1}{n+1}2^{-(2n+1)}} \sum_{n=0}^{(\varepsilon'-1)/2} \binom{2n+1}{n+1} 2^{-(2n+1)} \\ &= \frac{1}{\frac{3+\varepsilon'}{2+\varepsilon'}\frac{1}{2^{2+\varepsilon'}}\binom{2+\varepsilon'}{\frac{3+\varepsilon'}{2}}} \left(-1+2^{-2-\varepsilon'}(3+\varepsilon')\binom{2+\varepsilon'}{\frac{3+\varepsilon'}{2}}\right) \\ &= 2+\varepsilon' - \frac{2+\varepsilon'}{3+\varepsilon'} \cdot 2^{2+\varepsilon'} \Big/ \binom{2+\varepsilon'}{\frac{3+\varepsilon'}{2}}. \end{aligned} \tag{2.33}$$

Furthermore, the equation (2.30) follows from

$$\mathrm{Var}[L_\varepsilon] = G''_{L_\varepsilon}(1^-) + G'_{L_\varepsilon}(1^-) - (G'_{L_\varepsilon}(1^-))^2.$$

□

As already emphasized, $\{\rho_j\}_{j\geq 0}$ are the strict ascending ladder times. Thus, $\{R_j\}_{j\geq 0}$ are independent and identically distributed (iid) (see [51, XIII.1d, P.305]). Therefore, for the sake of a simplicity, instead of R_1 from now on we write R.

2.3.2 Brownian excursion limit for the simplified avalanche length

In order to obtain a first limit result the random variable $\frac{1}{n}R$, for $n \geq 1$, is considered. More precisely, the aim is to study the Laplace-Stieltjes transform for its distribution, namely $E[e^{-\frac{\lambda}{n}R}]$.

Lemma 2.5. *For fixed $\Re(\lambda) > 0$ the following asymptotics hold*

$$E[e^{-\frac{\lambda}{n}R}] = 1 - \sqrt{2\lambda} \cdot n^{-\frac{1}{2}} + \mathscr{O}\left(n^{-1}\right), \quad n \to \infty. \tag{2.34}$$

Proof. Elementary asymptotics, namely the exponential series $e^{-\lambda/n}$ as $n \to \infty$, and the power series of $\sqrt{1-z^2}$ as $z \to 0$, are developed to prove this lemma. More precisely, we have

$$e^{-\frac{\lambda}{n}} = 1 - \frac{\lambda}{n} + \frac{1}{2!}(\frac{\lambda}{n})^2 + \cdots \longrightarrow 1, \quad \textit{when} \quad n \to \infty, \tag{2.35}$$

and

$$\sqrt{1 - e^{2\frac{\lambda}{n}}} = \sqrt{2\lambda} \cdot n^{-\frac{1}{2}} + \mathscr{O}\left(n^{-1}\right), \quad \textit{when} \quad n \to \infty. \tag{2.36}$$

Therefore, by combining (2.19), (2.35) and (2.36), the following holds

$$E[e^{-\frac{\lambda}{n}R}] = \frac{1 - \sqrt{1 - e^{2\frac{\lambda}{n}}}}{e^{-\frac{\lambda}{n}}} = 1 - \sqrt{2\lambda} \cdot n^{-\frac{1}{2}} + \mathscr{O}\left(n^{-1}\right), \quad n \to \infty. \tag{2.37}$$

□

Thus, $\frac{1}{n}R$ converges in distribution to zero. This is not surprising, as it is essentially the length of the first excursion of the random walk. For the Brownian limit, this degenerates to zero, as arbitrary small excursions accumulate near time zero. But the second term in the asymptotic expansion is relevant for the simplified avalanche length. Another point of view can be established by connecting the limit to the Ito's excursion measure.

Some results from excursion theory In order to proceed further, some results from excursion theory are recalled, and this short overview of excursion theory is based on the theory comprehensively described in Revuz and Yor [101, XII.2, P.480]. Let $(e_t, t > 0)$ be an excursion process that takes values in a measurable space $(U, \mathscr{U})$. Furthermore, let us enhanced the measurable space $(U, \mathscr{U})$ by the set $\{\delta\}$ that corresponds to the zero-excursion, i.e. $U_\delta = U \cup \{\delta\}$. This set is naturally equipped with the σ-algebra $\mathscr{U}_\delta = \sigma(\mathscr{U}, \{\delta\})$.

For a measurable subset $\Gamma \subset \mathscr{U}_\delta$, the function

$$N_t^\Gamma(\omega) = \sum_{0<u\leq t} \mathbf{1}_\Gamma(e_u(\omega)) \tag{2.38}$$

is measurable.

The *Ito measure* n is the σ-finite measure defined on $\mathscr{U}$ by

$$n(\Gamma) := E\left[N_1^{\Gamma}\right] \tag{2.39}$$

and extended to $\mathscr{U}_\delta$ by $n(\delta) = 0$.

Since these function graphs are either entirely above or below the t-axis, we denote by U^+ and U^- the subsets of the set U. Furthermore, let n^+ and n^- be the upper and lower Ito measures, i.e. restrictions of n to U^+ and U^- respectively.

Following [101, XII.2, Theorem 2.4.], it turns out that the excursion process is a Poisson Point Process, and therefore its characteristic measure is the Ito measure. An important consequence of this is the Master Formula, see [101, XII, Proposition1.10] for its general version. Thus, by following [101, XII, Proposition1.10], the Master Formula states that for a positive $\mathscr{U}_\delta$-measurable function H defined on U_δ the following holds

$$E\left[H\left(e(\omega)\right)\right] = \int_U H(u)\, n(du). \tag{2.40}$$

Let $G(x)$ be the distribution of Brownian excursion lengths under the upper[2] Ito measure n_+. Then it is well-known, see [101, XII], that G has a density $g(x)$, that is defined as:

$$g(x) = \frac{1}{\sqrt{2\pi}} x^{-\frac{3}{2}}, \quad x > 0. \tag{2.41}$$

Furthermore, note that for every $\Re(\lambda) > 0$ following equality holds:

$$\int_0^\infty (1 - e^{-\lambda x}) g(x) dx = \sqrt{2\lambda}. \tag{2.42}$$

Proposition 2.2. *For $n \geq 1$ let $\mu_n = \mathrm{Law}(\frac{1}{n}R)$. Then, the following holds*

$$\lim_{n\to\infty} \int_0^\infty (1 - e^{-\lambda x}) n^{\frac{1}{2}} \mu_n(dx) = \sqrt{2\lambda}, \quad \Re(\lambda) > 0, \tag{2.43}$$

and thus $\lim_{n\to\infty} n^{\frac{1}{2}} \mu_n(dx) = G$ vaguely on $(0,\infty)$.

Proof. Let $\mu_n = \mathrm{Law}(\frac{1}{n}R)$ and ν denote the measure with density g. Then we can apply Lemma 5.1 and furthermore Proposition (5.1). Then, this proposition follows from (2.34).

[2] The distribution under the lower Ito measure n_- is the same.

□

In the following limit theorem ε is a strictly positive real number. For brevity instead of the more precise $L_{\lfloor n\varepsilon \rfloor}$ we write $L_{n\varepsilon}$. Let *erf* be the *error* function (see [90]), which is defined as:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \tag{2.44}$$

Theorem 2.1. *The scaled simplified avalanche length for the simple symmetric random walk converges in distribution to the simplified Brownian avalanche length. Analytically the following holds*

$$\lim_{n\to\infty} E[e^{-\frac{\lambda}{n} L_{n\varepsilon}}] = \frac{1}{\sqrt{\lambda\varepsilon\pi}\,\mathrm{erf}(\sqrt{\lambda\varepsilon}) + e^{-\lambda\varepsilon}}. \tag{2.45}$$

Proof. This formula can be derived from the Lévy measure of the subordinator consisting of Brownian passage times, see [[40], Theorem 16]. In [40] avalanche length L was defined as a random variable modeling the first time after which no orders get executed in an ε-time interval, where the orders are executed due to the price increase. Therefore, it corresponds to the simplified avalanche length in our model. Theorem 16 in [40] states that the Laplace transform of the avalanche length is given by

$$E\left[e^{-\lambda L}\right] = \frac{1}{\sqrt{\lambda\varepsilon\pi}\,\mathrm{erf}\left(\sqrt{\lambda\varepsilon}\right) + e^{-\lambda\varepsilon}}. \tag{2.46}$$

In order to derive the formula (2.46), author in [40] firstly considered the random variable L_y that is defined by conditioning L on $H = y$, $y \geq 0$, and H is height of an avalanche defined as the highest level at which a trade occurs during the period of avalanche.

Theorem 11 in [40] states that the avalanche height H is exponentially distributed with parameter $\sqrt{\frac{2}{\pi\varepsilon}}$, and thus

$$E\left[e^{-\lambda L}\right] = E\left[E\left[e^{-\lambda L}|H\right]\right] = \int_0^\infty E\left[e^{-\lambda L_y}\right] \sqrt{\frac{2}{\pi\varepsilon}} e^{-y\sqrt{\frac{2}{\pi\varepsilon}}} dy. \tag{2.47}$$

Theorem 14 in [40] established the Laplace transform of the L_y, and it is given by

$$E\left[e^{-\lambda L_y}\right] = \exp\left(-y\sqrt{\frac{2}{\pi\varepsilon}}(\sqrt{\lambda\varepsilon\pi}\,\mathrm{erf}\left(\sqrt{\lambda\varepsilon}\right) + e^{-\lambda\varepsilon} - 1)\right). \tag{2.48}$$

The Laplace transform of the avalanche length (2.46) is obtained by combining (2.47) and (2.48), and therefore the following equalities hold

$$\begin{aligned} E\left[e^{-\lambda L}\right] &= \int_0^\infty \exp\Big(-y\sqrt{\frac{2}{\pi\varepsilon}}(\sqrt{\lambda\varepsilon\pi}\,\mathrm{erf}\left(\sqrt{\lambda\varepsilon}\right)+e^{-\lambda\varepsilon}-1)\Big)\sqrt{\frac{2}{\pi\varepsilon}}e^{-y\sqrt{\frac{2}{\pi\varepsilon}}}dy \\ &= \sqrt{\frac{2}{\pi\varepsilon}}\int_0^\infty \exp\Big(-y\sqrt{\frac{2}{\pi\varepsilon}}(\sqrt{\lambda\varepsilon\pi}\,\mathrm{erf}\left(\sqrt{\lambda\varepsilon}\right)+e^{-\lambda\varepsilon})\Big)dy \\ &= \frac{1}{\sqrt{\lambda\varepsilon\pi}\,\mathrm{erf}\left(\sqrt{\lambda\varepsilon}\right)+e^{-\lambda\varepsilon}}. \end{aligned} \tag{2.49}$$

In the following, we derive equation (2.45) by excursion theory. Let $G(x)$ be the distribution of Brownian excursion lengths:

$$G(x)=\int_x^\infty g(y)\,dy, \quad \text{where} \quad g(y)=\frac{1}{\sqrt{2\pi}}y^{-3/2}. \tag{2.50}$$

We fix $n\in\mathbb{N}$ such that $n>1/\varepsilon$ and consider only excursions with length $R_i^{(n)}\in\left[\frac{1}{n},\varepsilon\right]$. Note that there exist only finitely many of those, i.e. $R_1^{(n)},\ldots,R_j^{(n),}$ and then there is a first excursion R_{j+1} (this one does not depend on n), whose length is more than ε. Define the modified avalanche length $L_{\varepsilon,n}$ by

$$L_{\varepsilon,n}=R_1^{(n)}+\ldots+R_j^{(n)}\le L_\varepsilon, \tag{2.51}$$

where L_ε is the simplified avalanche length, as defined by equation (2.14).

Since $\{R_i^{(n)}\}_{j\ge0}$ are iid, and there are a σ-discrete number of excursions, using the geometric series the following is obtained

$$E\left[e^{-\lambda L_{\varepsilon,n}}\right]=E\left[e^{-\lambda(R_1^{(n)}+\ldots+R_j^{(n)})}\right]=\frac{P(R>\varepsilon)}{1-E\left[e^{-\lambda R^{(n)}}\mathbb{I}_{R^{(n)}\le\varepsilon}\right]}. \tag{2.52}$$

By dominated convergence theorem, when $n\to\infty$, the following holds

$$E\left[e^{-\lambda L_\varepsilon}\right]=\frac{P(R>\varepsilon)}{1-E\left[e^{-\lambda R}\mathbb{I}_{R\le\varepsilon}\right]}. \tag{2.53}$$

In order to estimate the denominator

$$E\left[\mathbb{I}_{R\leq\varepsilon}\right]+E\left[\mathbb{I}_{R>\varepsilon}\right]-E\left[e^{-\lambda R}\mathbb{I}_{R\leq\varepsilon}\right]=E\left[\left(1-e^{-\lambda R}\right)\mathbb{I}_{R\leq\varepsilon}\right]+P\left(R>\varepsilon\right), \tag{2.54}$$

the function $\phi\left(x\right)$ is defined as

$$\phi\left(x\right)=\left(1-e^{-\lambda x}\right)\mathbb{I}_{x\leq\varepsilon}. \tag{2.55}$$

Thus, it holds:

$$\phi'\left(x\right)=\lambda e^{-\lambda x}\mathbb{I}_{x\leq\varepsilon}+\left(e^{-\lambda x}-1\right)\delta_{\varepsilon}\left(x\right). \tag{2.56}$$

Following the distribution of random variable R under the Ito measure n and by the Master Formula (2.40) (see [101, XII, Proposition1.10]) we have

$$\begin{aligned} E\left[\left(1-e^{-\lambda R}\right)\mathbb{I}_{R\leq\varepsilon}\right]&=\int\phi\left(R\left(u\right)\right)n\left(du\right)\\ E\left[\mathbb{I}_{R>\varepsilon}\right]&=\int\mathbb{I}_{R(u)>\varepsilon}n\left(du\right). \end{aligned} \tag{2.57}$$

The first equality in (2.58) is obtained by equation (2.57). Furthermore, we proceed by using Fubini's theorem, and then the third equality in (2.58) is obtained by (2.57). Thus, the following holds:

$$\begin{aligned} E\left[\left(1-e^{-\lambda R}\right)\mathbb{I}_{R\leq\varepsilon}\right]+E\left[\mathbb{I}_{R>\varepsilon}\right]&=\int\phi\left(R\left(u\right)\right)n\left(du\right)+\int\mathbb{I}_{R(u)>\varepsilon}n\left(du\right)\\ &=\int_0^{\infty}\phi'\left(x\right)G\left(x\right)dx+G\left(\varepsilon\right)\\ &=\lambda\int_0^{\varepsilon}e^{-\lambda x}G(x)\,dx+e^{-\lambda\varepsilon}G\left(\varepsilon\right)-G\left(\varepsilon\right)+G\left(\varepsilon\right)\\ &=\lambda\int_0^{\varepsilon}e^{-\lambda x}G(x)\,dx+e^{-\lambda\varepsilon}G\left(\varepsilon\right)\\ &=\left(e^{-\lambda\varepsilon}-1\right)G\left(\varepsilon\right)+\lambda\int_0^{\varepsilon}e^{-\lambda x}G(x)\,dx+\int_{\varepsilon}^{\infty}g(x)\,dx\\ &=\int_0^{\varepsilon}\left(1-e^{-\lambda x}\right)g(x)\,dx+\int_{\varepsilon}^{\infty}g(x)\,dx. \end{aligned} \tag{2.58}$$

The last equality in (2.58) is obtained by partial integration, i.e. $\int UdV=UV-\int VdU$ when $dV=g(x)dx$ and $U=1-e^{-\lambda x}$. Therefore, by combining (2.53) with obtained result

(2.58), we have

$$\frac{P(R>\varepsilon)}{1-E\left[e^{-\lambda R}\mathbb{I}_{R\leq\varepsilon}\right]}=\frac{G(\varepsilon)}{\int_0^\varepsilon\left(1-e^{-\lambda x}\right)g(x)\,dx+\int_\varepsilon^\infty g(x)\,dx}, \tag{2.59}$$

where $G(\varepsilon)=\int_\varepsilon^\infty g(x)\,dx=\int_\varepsilon^\infty\frac{1}{\sqrt{2\pi}}x^{-3/2}\,dx=\frac{\sqrt{2}}{\sqrt{\pi\varepsilon}}$.

The required formula is obtained by a straightforward computation. □

Corollary 2.5.

$$\lim_{n\to\infty}E[\frac{1}{n}L_{n\varepsilon}]=\varepsilon,\quad\lim_{n\to\infty}\mathrm{Var}[\frac{1}{n}L_{n\varepsilon}]=\frac{4}{3}\varepsilon^2. \tag{2.60}$$

Proof. From differentiating the generating function. □

We note that the moments from Corollary 2.5 are consistent with the asymptotics of the moments from Corollary 2.4.

Remark 2.10. *The distribution of the simplified Brownian avalanche length was obtained by [40]. We see that the limit of the random walk avalanche length agrees with the Brownian avalanche length. To prove that the limit distribution actually is the Brownian avalanche length distribution, without referring to the result in [40], would require further justification, perhaps a continuity argument for the avalanche length as a functional on the Skorohod (or Wiener) space. This is postponed for now.*

The analytical result above allows us to study the distribution of L_ε in more detail.

2.4 The full avalanche length

In this section, we study the full avalanche length $L^*_{\mu,\varepsilon}$ with parameter $\mu\geq 1$ and window parameter $\varepsilon>0$. The full avalanche continues until there is a time interval of length greater then ε that contains neither Type I nor Type II trades.

If $T_1\leq\varepsilon,\ldots,T_k\leq\varepsilon,T_{k+1}>\varepsilon$, for $k\geq 1$, the full avalanche length $L^*_{\mu,\varepsilon}$, $\mu\geq 1,\varepsilon>0$, is formally defined as follows

$$L^*_{\mu,\varepsilon}=T_1+\ldots+T_k. \tag{2.61}$$

Thus, the full avalanche length is obtained from a sequence of trades with intertrading time less than ε followed by a trade with intertrading time bigger than ε.

2.4.1 The generating functions for the full avalanche length

Let us recall the definition established in equation (2.11). Note that $(s_0,\dots,s_n)$ is a path of the *size* $n \geq 1$, and $\mathbf{U}^{(n)}$ is defined as

$$\mathbf{U}^{(n)} = \left\{(s_0,\dots,s_n) \in \mathbb{Z}^{n+1} : s_0 = 0,\, |s_j - s_{j-1}| = 1 \text{ for } j = 1,\dots,n\right\}.$$

Given the parameter $\mu \geq 1$ three sets of random walk paths of length $n \geq 1$ are defined as follows

$$\mathscr{A}_{n,\mu} = \left\{s \in U_n : s_0 = 0,\, -\mu < s_k \leq 0 \text{ for } 0 < k \leq n-1 \text{ and } s_n = +1\right\}, \tag{2.62}$$

$$\mathscr{B}_{n,\mu} = \left\{s \in U_n : s_0 = 0,\, -\mu < s_k \leq 0 \text{ for } 0 \leq k \leq n-1,\, s_{n-1} = 0 \text{ and } s_n = -1\right\}, \tag{2.63}$$

$$\mathscr{C}_{n,\mu} = \left\{s \in \mathscr{A}_{n,\mu} :\ \min(s_0,\dots,s_{n-1}) = -\mu+1\right\}. \tag{2.64}$$

Furthermore, the corresponding sets with arbitrary, finite length paths, are defined as

$$\mathscr{A}_\mu = \bigcup_{n\geq 1} \mathscr{A}_{n,\mu}, \quad \mathscr{B}_\mu = \bigcup_{n\geq 1} \mathscr{B}_{n,\mu}, \quad \mathscr{C}_\mu = \bigcup_{n\geq 1} \mathscr{C}_{n,\mu}. \tag{2.65}$$

Examples of the paths that belongs to the sets $\mathscr{A}_\mu$, $\mathscr{B}_\mu$ and $\mathscr{C}_\mu$ are given in Figure 2.8, Figure 2.9 and Figure 2.10 respectively.

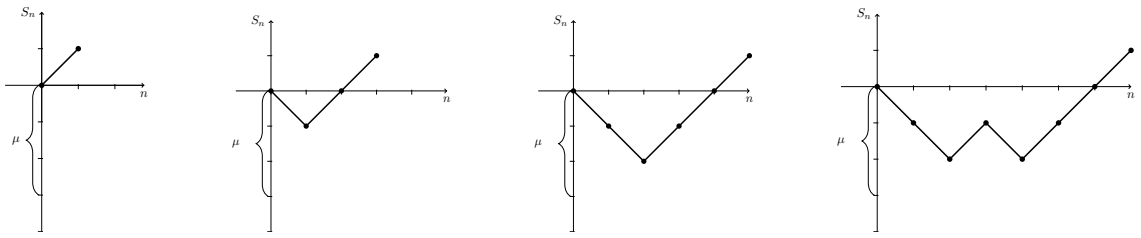

Figure 2.8: *Some elements of* $\mathscr{A}_\mu$

Note that the corresponding probability generating functions are obtained from the ordinary generating functions by the substitution $z \mapsto z/2$, since the introduced model is

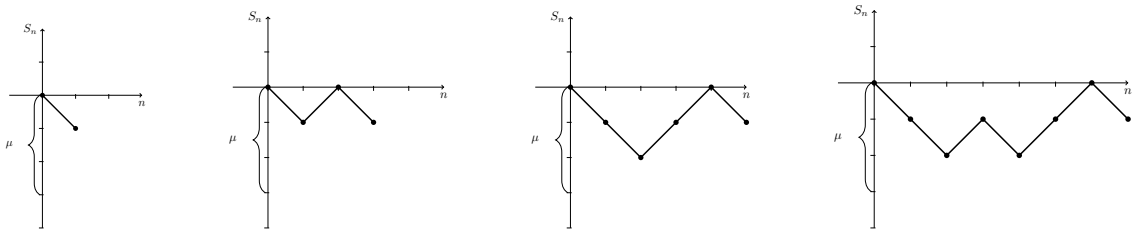

Figure 2.9: *Some elements of* $\mathscr{B}_\mu$

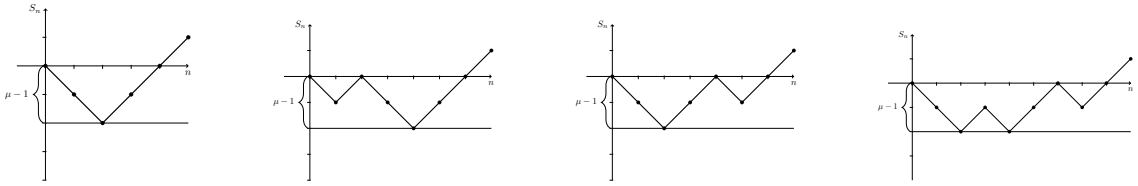

Figure 2.10: *Some elements of* $\mathscr{C}_\mu$

the symmetric Bernoulli model. As in [51, XIV.4] let us introduce

$$\lambda_1(s) = \frac{1+\sqrt{1-s^2}}{s}, \quad \lambda_2(s) = \frac{1-\sqrt{1-s^2}}{s}. \tag{2.66}$$

Then the following lemma states the probability generating functions for classes $\mathscr{A}_\mu, \mathscr{B}_\mu, \mathscr{C}_\mu$.

Lemma 2.6. *The probability generating functions for combinatorial classes* $\mathscr{A}_\mu, \mathscr{B}_\mu, \mathscr{C}_\mu$ *are given by*

$$A_\mu(s) = \frac{\lambda_1(s)^\mu - \lambda_2(s)^\mu}{\lambda_1(s)^{\mu+1} - \lambda_2(s)^{\mu+1}}, \tag{2.67}$$

$$B_\mu(s) = A_\mu(s), \tag{2.68}$$

$$C_\mu(s) = A_\mu(s) - A_{\mu-1}(s). \tag{2.69}$$

Proof. Shifting a path from $\mathscr{A}_\mu$ by μ we obtain a path corresponding to absorption at $\mu+1$ at the n-th trial in a symmetric game of gamblers ruin with initial position μ and forbidden barrier at 0. The generating function for the probability of the ruin (absorption at 0, forbidden level $a > 0$, and initial position $0 < z < a$) at the $n-$th step is derived in [51, XIV.4, (4.12), P.351], and it is

$$\frac{\lambda_1(s)^z - \lambda_2(s)^z}{\lambda_1(s)^a - \lambda_2(s)^a}. \tag{2.70}$$

Using formula (2.70) for $p = 1/2$, $q = 1/2$, $z = \mu$, $a = \mu + 1$ yields (2.67). The last step for a path in $\mathscr{B}_{n,\mu}$ is down. If it is replaced by an up-step, a path in $\mathscr{A}_{n,\mu}$ is obtained. Thus we have a bijection from $\mathscr{B}_{n,\mu}$ to $\mathscr{A}_{n,\mu}$ yielding (2.68). The set $\mathscr{C}_{n,\mu}$ contains paths of length n, that hit the level $-\mu+1$, but they do not hit the level $-\mu$, thus

$$\mathscr{C}_{n,\mu} = \mathscr{A}_{n,\mu} \setminus \mathscr{A}_{n,\mu-1}. \tag{2.71}$$

Therefore, since $\mathscr{A}_{n,\mu-1} \subseteq \mathscr{A}_{n,\mu}$ the equation (2.69) holds true. □

The numbers of paths of length $n = 1,...,10,11$ that belong to the class $\mathscr{A}_\mu$, when $\mu = 1,...,n$, are established in Table 5.3.3 (see Appendix 5.3.3).

The following Proposition 2.3 states the key result describing Type II trades. More precisely, Proposition 2.3 establishes a path decomposition for trading excursions that lead to a Type II trade, which is, in fact, related to the decomposition illustrated in [101, Sec.VI.3, Fig.5,P.256]. Firstly, we state the path decomposition and the proof, which will refer to a few auxiliary lemmas. For the clarity of exposition, those auxiliary lemmas are provided afterward at the end of this subsection.

Proposition 2.3. *Suppose that the first trade is a Type II. Then there exists an integer* $K \geq 1$, *such that* $(S(0),\ldots,S(T_1))$ *is the concatenation of* K *elements from* $\mathscr{B}_\mu$ *and one element from* $\mathscr{C}_\mu$.

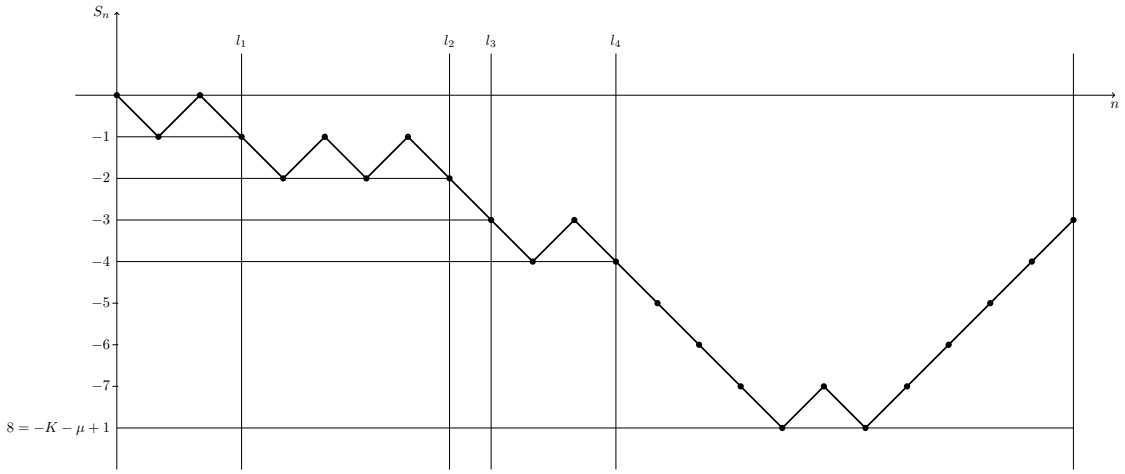

Figure 2.11: *Path decomposition of a Type II trading excursion,* $\mu = 5$, $K = 4$

Proof. Assume that the first trade is Type II. Then $(S(0),...,S(T_1))$ contains no Type I trade, thus $S(0),...,S(T_1)$ are non positive by Lemma 2.9. A Type II trade occurs when the price goes down by μ or more steps and then goes up by μ steps. The first time this behavior is completed, the excursion ends.

Let $K = -S(T_1 - 1)$ and $\ell^* = \max\{0 \le n < T_1 - 1 : S(n) = -K\}$, see Figure 2.11 for an illustration for $\mu = 5$, $K = 4$ and $\ell^* = \ell_4$. Since the Type II trade is completed, the path $(S(\ell^*),...,S(T_1))$ must go down to level $-K-\mu+1$, and at that moment the new order is placed at level $-K+1$. Then, the Type II trade occurs at time T_1, i.e. $S(T_1) = -K+1$. Note that if the path $(S(\ell^*),...,S(T_1))$ did go deeper, a Type II trade would be followed by a Type I trade which would complete before T_1. Thus this part of the path, namely the path $(S(\ell^*),...,S(T_1))$, belongs to the class $\mathscr{C}_\mu$.

Note that every level from 0 to $-K$ is visited by $(S(0),...,S(\ell^*))$. Let $\ell_0 = 0$ and let ℓ_k be the *last* time level $-k$ is visited before ℓ^*+1 where $k = 1,...,K$, i.e.

$$\ell^k = \max\{0 < n < \ell^* + 1 : S(n) = -k\}.$$

Since there is no trade during this initial period, and there is no Type I trade in particular, by Lemma 2.8, the paths $(S(\ell_{j-1}),...,S(\ell_j))$, for $j = 1,...,K$, have depth less than μ. Therefore, there are K paths, precisely $(S(\ell_{j-1}),...,S(\ell_j))$ $j = 1,...,K$, that belong to class $\mathscr{B}_\mu$. □

In the Definition 2.5, which is the Definition 4.1. in [6], $\mathscr{F} = (\mathscr{F}_t)_{t\ge 0}$ denotes a filtration satisfying the usual conditions on a filtered probability space $(\Omega, \mathscr{F}, P)$. Furthermore, for a definition of honest times see [[71], p. 73].

Definition 2.5. A random time τ is an $\mathbb{F}$-honest time if for every $t > 0$ there exists an $\mathscr{F}_t$-measurable random variable τ_t such that $\tau_t = \tau$ on $\{\tau < t\}$. Then, it is always possible to choose τ_t such that $\{\tau_t \le t\}$.

Remark 2.11. *The ℓ_j are not stopping times, but honest times.*

Remark 2.12. *The final down segments in the $\mathscr{B}_\mu$ paths seem to vanish in the Brownian limit, but they correspond to the support of the local time at the minimum.*

Lemma 2.7. *If the first trade is a Type II trade, then*

$$\max\{S(0),\dots,S(T_1)\} \le 0, \quad \min\{S(0),\dots,S(T_1)\} \le -\mu. \tag{2.72}$$

Proof. Assume by contradiction that there is a time $n \in [0, T_1]$ such that $S_n > 0$. Furthermore, assume that this time n is the minimum time with this property. Then $V(n, S_n) > 0$ and $S_n > S_0$, thus the first trade would be a Type I trade. Assume next by contradiction that $-\mu < S_n \leq 0$ for all $n \in [0, T_1]$. Since $\alpha_0 = 0$ and $S_0 = 0$, then $\alpha_1 = 1$. Then the recursion for α_n shows $\alpha_n = 1$ (since $\alpha_n \neq S_n$ and $\alpha_n \neq S_n + \mu + 1$) for all $n \in [1, T_1]$, and no trade could take place. □

Lemma 2.8. *Suppose $0 \leq a \leq b$ and S(a)=S(b). If*

$$\min\{S(a), \ldots, S(b)\} < S(a) - \mu, \tag{2.73}$$

there is a Type II trade followed by a Type I trade in $[a, b]$. Moreover, if

$$\min\{S(a), \ldots, S(b)\} = S(a) - \mu, \tag{2.74}$$

a Type II trade occurs at time b.

Proof. Let

$$\gamma = \min(S(a), \ldots, S(b), \quad \beta = \min\{n \in [a, b] : S(n) = \gamma\}, \tag{2.75}$$

and

$$\delta = \min\{n \in [\beta, b] : S(n) = \gamma + \mu\}. \tag{2.76}$$

At time β a new order is placed at the price level $S_n + \mu = \gamma + \mu$, and therefore we have $V(\beta + 1, \gamma + \mu) > 0$. This order at price level $\gamma + \mu$ gets executed at time δ.

The second part of the lemma follows directly from the definition of Type II trades.

□

Lemma 2.9. *The first trade after zero is a Type I trade if and only if*

$$\max\{S(0), \ldots, S(T_1 - 1)\} = 0, \quad S(T_1) = 1 \tag{2.77}$$

and

$$\min(S_0, \ldots, S(T_1 - 1)) > -\mu. \tag{2.78}$$

Proof. Suppose that the first trade is a Type I trade. Then there is no Type II trade during the time interval $[0, T_1]$, and thus by Lemma 2.8 inequality (2.78) holds. Note that the

price level 1 is the strict ladder time, and therefore, according to Lemma 2.1, a Type I trade occurs at the price level 1. Since the first trade occurs at time T_1, it directly implies that (2.77) holds. Conversely, assume that (2.78) and (2.77) hold. Then, by Lemma 2.1 a Type I trade ocurs at time T_1, and there is no earlier Type I trade. Furthermore, Lemma 2.7 shows there is no Type II trade earlier. □

Proposition 2.4. *The probability generating function of the time to the next trade $T_\mu(z)$ is given by*

$$T_\mu(z) = E[z^{T_1}] = A_\mu(z) + \frac{B_\mu(z)}{1 - B_\mu(z)} \cdot C_\mu(z). \tag{2.79}$$

Proof. Let us define the set $\mathscr{T}_{\mu,n}$ of all paths where the first trade occurs at step n. and $\mathscr{T}_\mu = \bigcup_{n\geq 1} \mathscr{T}_{\mu,n}$. This set $\mathscr{T}_\mu$ contains elements that are either a path of a Type I trade or a path of a Type II trade. By Lemma 2.9 a path of a Type I trade belongs to the class $\mathscr{A}_\mu$. By the path decomposition presented in Proposition 2.3, a path of a Type II trade corresponds bijectivly to the Cartesian product of the set of finite sequences of length 1 or more of elements in $\mathscr{B}_\mu$ times an element from the class $\mathscr{C}_\mu$. Thus, using the symbolic notation from [53, Section I] this is written more clearly as

$$\mathscr{T}_\mu = \mathscr{A}_\mu \cup \mathrm{SEQ}_{\geq 1}(\mathscr{B}_\mu) \times \mathscr{C}_\mu. \tag{2.80}$$

From [53, Theorem I.1, P.27] and the section on *restricted constructions* (in particular $\mathrm{SEQ}_{\geq k}$ in [53, P.30]), (2.79) directly follows from (2.80). □

Once the law of T_1 is found, the focus is on the computing distribution of the avalanche length.

Theorem 2.2. *The full avalanche length for the symmetric random walk has probability generating function*

$$E[z^{L^*_{\mu\varepsilon}}] = \frac{P[T_1 > \varepsilon]}{E[1 - z^{T_1}; T_1 \leq \varepsilon] + P[T_1 > \varepsilon]}, \tag{2.81}$$

where T_1 has the probability generating function given in (2.79) above.

Proof. The fact that $(T_i)_{i\geq 1}$ are independent and identically distributied and equation (2.61),

yield that

$$\begin{aligned}
E[z^{L^*_{\mu\varepsilon}}] &= \sum_{k\geq 0} E[z^{T_1+\cdots+T_k} : T_1 \leq \varepsilon, T_2 \leq \varepsilon, ..., T_k \leq \varepsilon, T_{k+1} > \varepsilon] \\
&= \sum_{k\geq 0} E[z^{T_1} z^{T_2} \cdots z^{T_k} : T_1 \leq \varepsilon, T_2 \leq \varepsilon, ..., T_k \leq \varepsilon, T_{k+1} > \varepsilon] \\
&= \sum_{k\geq 0} E[z^{T_1} : T_1 \leq \varepsilon]^k E[1 : T_{k+1} > \varepsilon] = \frac{P[T_{k+1} > \varepsilon]}{1 - E[z^{T_1} : T_1 \leq \varepsilon]} \qquad (2.82) \\
&= \frac{P[T_{k+1} > \varepsilon]}{E[1 : T_1 \leq \varepsilon] + E[1 : T_1 > \varepsilon] - E[z^{T_1} : T_1 \leq \varepsilon]} \\
&= \frac{P[T_{k+1} > \varepsilon]}{E[1 - z^{T_1} : T_1 \leq \varepsilon] + P[T_1 > \varepsilon]} = \frac{P[T_1 > \varepsilon]}{E[1 - z^{T_1} : T_1 \leq \varepsilon] + P[T_1 > \varepsilon]}.
\end{aligned}$$

□

In addition, the following Lemma 2.10 presents the probability generating functions of some interesting quantities. More precisely, the following quantities are considered: the time to the first trade assuming it is a Type I trade, the time to the first trade assuming it is a Type II trade, and the time to the first Type II trade.

Denote by D the index of the first Type II trade in the sequence of all trades, i.e.

$$D = \inf\{i \geq 1 : S(\tau_i) < S(\tau_{i-1})\}. \qquad (2.83)$$

Consequently, the time to the first Type II trade is τ_D.

Lemma 2.10. *The following equalities hold*[3]

$$E[z^{T_1}; S(T_1) > 0] = A_\mu(z), \quad E[z^{T_1}; S(T_1) \leq 0] = \frac{B_\mu(z)}{1 - B_\mu(z)} \cdot C_\mu(z) \qquad (2.84)$$

and

$$E[z^{\tau_D}] = \frac{1}{1 - E[z^{T_1}; S(T_1) > 0]} \cdot E[z^{T_1}; S(T_1) \leq 0]. \qquad (2.85)$$

Proof. By following Lemma 2.9 a Type I trading excursion corresponds to an element in $\mathscr{A}_\mu$, and therefore by [53, Theorem I.1, P.27] the left equation in (2.84) holds. By Proposition 2.3 a path of a Type II trading excursion corresponds to the second term on the right hand side of (2.80), and thus by [53, Theorem I.1, P.27] the right equation

[3] We use semicolon notation, $E[X;A] = E[XI_A]$ for an integrable random variable X and an event A.

in (2.84) holds. In particular, the path to the next Type II trades consists of a (possible empty) sequence of Type I trading excursions followed by a Type II trading excursion. Therefore, by applying [53, Theorem I.1, P.27] the corresponding generating functions are obtained. □

Corollary 2.6.

$$P[S(T_1) > 0] = \frac{\mu}{\mu+1} \tag{2.86}$$

Proof. By sending $s \to 1$ in the (2.84) we have:

$$P[S(T_1) > 0] = \lim_{s\uparrow 1} A_\mu(s).$$

The following equalities hold

$$\begin{aligned}
\lambda_1(s)^\mu - \lambda_2(s)^\mu &= \left(\frac{1+\sqrt{1-s^2}}{s}\right)^\mu - \left(\frac{1-\sqrt{1-s^2}}{s}\right)^\mu \\
&= s^{-\mu} \sum_{k=0}^{\mu} \left[\binom{\mu}{k}(1-s^2)^{\frac{k}{2}} - \binom{\mu}{k}(-1)^k(1-s^2)^{\frac{k}{2}}\right] \\
&= s^{-\mu} \sum_{k=0}^{\lfloor\frac{\mu}{2}\rfloor-1} 2\left[\binom{\mu}{2k+1}(1-s^2)^{\frac{2k+1}{2}}\right],
\end{aligned} \tag{2.87}$$

and thus

$$\begin{aligned}
\frac{\lambda_1(s)^\mu - \lambda_2(s)^\mu}{\lambda_1(s)^{\mu+1} - \lambda_2(s)^{\mu+1}} &= s\frac{\sum_{k=0}^{\lfloor\frac{\mu}{2}\rfloor-1}\left[\binom{\mu}{2k+1}(1-s^2)^{\frac{2k+1}{2}}\right]}{\sum_{k=0}^{\lfloor\frac{\mu+1}{2}\rfloor-1}\left[\binom{\mu+1}{2k+1}(1-s^2)^{\frac{2k+1}{2}}\right]} \\
&= s\frac{\binom{\mu}{1}(1-s^2)^{\frac{1}{2}} + \sum_{k=1}^{\lfloor\frac{\mu}{2}\rfloor-1}\left[\binom{\mu}{2k+1}(1-s^2)^{\frac{2k+1}{2}}\right]}{\binom{\mu+1}{1}(1-s^2)^{\frac{1}{2}} + \sum_{k=1}^{\lfloor\frac{\mu+1}{2}\rfloor-1}\left[\binom{\mu+1}{2k+1}(1-s^2)^{\frac{2k+1}{2}}\right]}.
\end{aligned} \tag{2.88}$$

Then, we obtain

$$\begin{aligned}
A_\mu(1) &= \lim_{s\uparrow 1} \frac{\lambda_1(s)^\mu - \lambda_2(s)^\mu}{\lambda_1(s)^{\mu+1} - \lambda_2(s)^{\mu+1}} \\
&= \lim_{s\uparrow 1} s\frac{\binom{\mu}{1}(1-s^2)^{\frac{1}{2}} + \sum_{k=1}^{\lfloor\frac{\mu}{2}\rfloor-1}\left[\binom{\mu}{2k+1}(1-s^2)^{\frac{2k+1}{2}}\right]}{\binom{\mu+1}{1}(1-s^2)^{\frac{1}{2}} + \sum_{k=1}^{\lfloor\frac{\mu+1}{2}\rfloor-1}\left[\binom{\mu+1}{2k+1}(1-s^2)^{\frac{2k+1}{2}}\right]} = \frac{\mu}{\mu+1}.
\end{aligned} \tag{2.89}$$

μ \ n	1	2	3	4	5	6	7	8	9	10
1	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512	1/1024
2	1/2	0	1/8	1/16	1/16	3/64	5/128	1/32	13/512	21/1024
3	1/2	0	1/8	0	1/16	1/64	5/128	5/256	7/256	19/1024
4	1/2	0	1/8	0	1/16	0	5/128	1/256	7/256	7/1024
5	1/2	0	1/8	0	1/16	0	5/128	0	7/256	1/1024
6	1/2	0	1/8	0	1/16	0	5/128	0	7/256	0
7	1/2	0	1/8	0	1/16	0	5/128	0	7/256	0

Table 2.1: *The probabilities* $p_n = P[T_1 = n]$ *for* $n = 1,...,10$ *and* $\mu = 1,...,7$

□

2.4.2 Computing probabilities for $\mu = 1,...,7$

Combining (2.79) with (2.67), (2.68) and (2.69), entails that

$$T_\mu(z) = A_\mu(z) + \frac{A_\mu(z)}{1-A_\mu(z)}(A_\mu(z) - A_{\mu-1}(z)) = \frac{A_\mu - A_\mu(z)A_{\mu-1}(z)}{1-A_\mu(z)}. \tag{2.90}$$

For $\mu = 1,2,3,4,5,6,7$ series expansions are used to expand the corresponding formula (2.90). Then, the probabilities $p_n = P[T_1 = n]$ are calculated for $n = 1,...,10$, and values are established in Table 2.1. Furthermore, the following holds true:

$$\begin{aligned} T_\mu(z) &= p_0 + \sum_{n\geq 1}\Big[\sum_{k=0}^{n} p_k z^n - \sum_{k=0}^{n-1} p_k z^n\Big] = p_0 + \sum_{n\geq 1}\sum_{k=0}^{n} p_k z^n - \sum_{n\geq 1}\sum_{k=0}^{n-1} p_k z^n \\ &= \sum_{n\geq 0}\sum_{k=0}^{n} p_k z^n - \sum_{n\geq 0}\sum_{k=0}^{n} p_k z^{n+1} = \sum_{n\geq 0}(\sum_{k=0}^{n} p_k) z^n (1-z). \end{aligned} \tag{2.91}$$

Let $q_n = P[T_1 > n]$, i.e. $q_n = \sum_{k\geq n+1} P[T_1 = k]$. Further, let $Q_\mu(z) = \sum_{n\geq 0} q_n z^n$, and thus:

$$Q_\mu(z) = \sum_{n\geq 0}(\sum_{k\geq n+1} p_k) z^n = \frac{1-T_\mu(z)}{1-z}.$$

See Table 2.2 for values of probabilities q_ε, when $\varepsilon = 1,...,10,11$ and $\mu = 1,...,7$.

μ \ ε	1	2	3	4	5	6	7	8	9
1	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512
2	1/2	1/2	3/8	5/16	1/4	13/64	21/128	17/128	55/512
3	1/2	1/2	3/8	3/8	5/16	19/64	33/128	61/256	27/128
4	1/2	1/2	3/8	3/8	5/16	5/16	35/128	69/256	31/128
5	1/2	1/2	3/8	3/8	5/16	5/16	35/128	35/128	63/256
6	1/2	1/2	3/8	3/8	5/16	5/16	35/128	35/128	63/256
7	1/2	1/2	3/8	3/8	5/16	5/16	35/128	35/128	63/256

Table 2.2: *The probabilities* $q_\varepsilon = P[T_1 > \varepsilon]$ *for* $\varepsilon = 1,...,9$ *and* $\mu = 1,...,7$

Let us recall (2.4.1), i.e.

$$E[z^{L^*_{\mu\varepsilon}}] = \frac{P[T_1 > \varepsilon]}{1 - E[z^{T_1} : T_1 \leq \varepsilon]} = \frac{q_\varepsilon}{1 - \sum_{k=1}^{\varepsilon} z^k p_k}. \tag{2.92}$$

Plugging values from Table 2.1 and Table 2.2, when $\mu = 1,2,3,4$ and $\varepsilon = 1,2,3,4,5$, in equation (2.92), the probability generating functions of the full avalanche length $L^*_{\mu,\varepsilon}$ can be derived (see Table 2.3).

μ \ ε	1	2	3	4	5
1	$\frac{1}{2-z}$	$\frac{1}{4-2z-z^2}$	$\frac{1}{8-4z-2z^2-z^3}$	$\frac{1}{16-8z-4z^2-2z^3-z^4}$	$\frac{1}{32-16z-8z^2-4z^3-2z^4-z^5}$
2	$\frac{1}{2-z}$	$\frac{1}{2-z}$	$\frac{3}{8-4z-z^3}$	$\frac{5}{16-8z-2z^3-z^4}$	$\frac{4}{16-8z-2z^3-z^4-z^5}$
3	$\frac{1}{2-z}$	$\frac{1}{2-z}$	$\frac{3}{8-4z-z^3}$	$\frac{3}{8-4z-z^3}$	$\frac{5}{16-8z-2z^3-z^5}$
4	$\frac{1}{2-z}$	$\frac{1}{2-z}$	$\frac{3}{8-4z-z^3}$	$\frac{3}{8-4z-z^3}$	$\frac{5}{16-8z-2z^3-z^5}$

Table 2.3: *The probability generating function of the full avalanche length* $L^*_{\mu,\varepsilon}$ *when* $\mu = 1,2,3,4$ *and* $\varepsilon = 1,2,3,4,5$

ε \ k	1	2	3	4	5	6	7	8
1	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512
2	1/8	1/8	3/32	5/64	1/16	13/256	21/512	17/512
3	1/16	1/16	1/16	7/128	13/256	3/64	11/256	81/2048
4	1/32	1/32	1/32	1/32	15/512	29/1024	7/256	27/1024
5	1/64	1/64	1/64	1/64	1/64	31/2048	61/4096	15/1024

Table 2.4: *The probabilities* $P[L^*_{1,\varepsilon} = k]$, *for* $k = 1,...,8$, $\mu = 1$ *and* $\varepsilon = 1,2,3,4,5$.

ε \ k	1	2	3	4	5	6	7	8
1	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512
2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512
3	3/16	3/32	3/32	9/128	3/64	9/256	27/1024	39/2048
4	5/32	5/64	5/64	5/64	15/256	45/1024	75/2048	125/4096
5	1/8	1/16	1/16	1/16	1/16	13/256	21/512	37/1024

Table 2.5: *The probabilities* $P[L^*_{1,\varepsilon} = k]$, *for* $k = 1,...,8$, $\mu = 2$ *and* $\varepsilon = 1,2,3,4,5$.

ε \ k	1	2	3	4	5	6	7	8
1	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512
2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512
3	3/16	3/32	3/32	9/128	3/64	9/256	27/1024	39/2048
4	3/16	3/32	3/32	9/128	3/64	9/256	27/1024	39/2048
5	5/32	5/64	5/64	15/256	15/256	25/512	75/2048	125/4096

Table 2.6: *The probabilities* $P[L^*_{1,\varepsilon} = k]$, *for* $k = 1,...,8$, $\mu = 3,4$ *and* $\varepsilon = 1,2,3,4,5$.

Furthermore, the probability generating functions of the full avalanche lengths, which are presented in Table 2.3, can be expanded. Thus, probabilities that the full avalanche length takes particular values are calculated and summarized in Table 2.4 (for $\mu = 1$), Table 2.5 (for $\mu = 2$) and Table 2.6 (for $\mu = 3$ and $\mu = 4$).

2.4.3 Limit results for the full avalanche length

If we consider the Brownian limit in distribution on the Skorohod space we have

$$\left\{ \frac{1}{\sqrt{n}} S_{\lfloor nt \rfloor} : t \geq 0 \right\} \to \{W(t) : t \geq 0\}, \tag{2.93}$$

see [14, Theorem 14.1. of XIV,P.146]. Furthermore, the linear interpolation on the Wiener space can be derived similarly, see [14, Theorem 8.6. of VIII,P.90]. Note that time must be scaled linearly, i.e. $t \mapsto nt$, and space must be scaled with the square-root, i.e. $\mu \mapsto \sqrt{n}\mu$. The excursion length refers to time, while the excursion depth refers to space.

Up to here, in the notation for intertrading times the dependence on the parameter μ has been omitted. Since μ must be used as a variable in the limiting procedure, from now on we write $T_1^{(\mu)}$ instead of T_1, and for brevity $T_1^{(n\mu)}$ instead of the more precise $T_1^{(\lfloor n\mu \rfloor)}$.

Proposition 2.5. *For the Laplace-Stieltjes transform of a random variable* $\frac{1}{n} T_1^{(\sqrt{n}\mu)}$ *we have*

$$E[e^{-\frac{s}{n} T_1^{(\sqrt{n}\mu)}}] = 1 - \sqrt{2s} \tanh\left(\mu\sqrt{2s}\right) \cdot n^{-\frac{1}{2}} + \mathscr{O}\left(n^{-1}\right), \quad n \to \infty. \tag{2.94}$$

Proof. Slightly lengthy, but still elementary asymptotic[4] are employed to prove this result. Firstly, by using the expansion of $\sqrt{1-z^2}$ for $z \to 1$, the expansion of $\lambda_1(z)$ obtained, namely

$$\lambda_1(z) - \lambda_1(1) = \lambda_1(z) - 1 = \sqrt{2}(1-z)^{1/2} + (1-z) + \frac{3}{2\sqrt{2}}(1-z)^{3/2} + \mathscr{O}(1-z)^2.$$

Furthermore, inserting the $\lambda_1(z) - 1$ into the power series for $\log(1+x)$ at $x = 0$, yields

$$\log \lambda_1(z) = \log(\lambda_1(z) - 1 + 1) = \sqrt{2}(1-z)^{1/2} + \frac{5}{6\sqrt{2}}(1-z)^{3/2} + \mathscr{O}(1-z)^2.$$

A similar calculation for $\lambda_2(z)$ yields

$$\lambda_2(z) - 1 = -\sqrt{2}(1-z)^{1/2} + (1-z) + \frac{3}{2\sqrt{2}}(1-z)^{3/2} + \mathscr{O}(1-z)^2.$$

[4] The detailed calculations can be performed by hand in a few pages but can be also performed (and checked) automatically with a computer algebra system.

$$\log \lambda_2(z) = \log(\lambda_2(z) - 1 + 1) = -\sqrt{2}(1-z)^{1/2} - \frac{5}{6\sqrt{2}}(1-z)^{3/2} + \mathcal{O}(1-z)^2.$$

So, the following asymptotics hold

$$\log \lambda_1(e^{-s/n}) = \sqrt{2}(1-e^{-s/n})^{1/2} + \frac{5}{6\sqrt{2}}(1-e^{-s/n})^{3/2} + \mathcal{O}(1-e^{-s/n})^2, \tag{2.95}$$

$$\log \lambda_2(e^{-s/n}) = -\sqrt{2}(1-e^{-s/n})^{1/2} - \frac{5}{6\sqrt{2}}(1-e^{-s/n})^{3/2} + \mathcal{O}(1-e^{-s/n})^2. \tag{2.96}$$

By expanding $e^{-s/n}$, when $n \to \infty$, using the exponential series, we know

$$1 - e^{-s/n} = \frac{s}{n} - \frac{s^2}{2n^2} + \mathcal{O}(n^{-3}), \quad n \to \infty.$$

Thus, the following equalities hold

$$(1-e^{-s/n})^{1/2} = \frac{s^{1/2}}{n^{1/2}}\left[1 - \frac{s}{2n} + \mathcal{O}(n^{-2})\right]^{1/2} = \frac{s^{1/2}}{n^{1/2}}\left[1 - \frac{s}{4n} + \mathcal{O}(n^{-2})\right], \tag{2.97}$$

$$(1-e^{-s/n})^{3/2} = \frac{s^{3/2}}{n^{3/2}}\left[1 - \frac{s}{2n} + \mathcal{O}(n^{-2})\right]^{3/2} = \frac{s^{3/2}}{n^{3/2}}\left[1 - \frac{3s}{4n} + \mathcal{O}(n^{-2})\right]. \tag{2.98}$$

Furthermore, we combine the asymptotic expansions (2.95), (2.97) and (2.98), with the elementary formula

$$\lambda_1(e^{-s/n})^{\mu\sqrt{n}} = \exp(\mu\sqrt{n}\log\lambda_1(e^{-s/n})). \tag{2.99}$$

Similarly, the asymptotic expansions (2.96), (2.97) and (2.98) are combined with the formula

$$\lambda_2(e^{-s/n})^{\mu\sqrt{n}} = \exp(\mu\sqrt{n}\log\lambda_2(e^{-s/n})). \tag{2.100}$$

Equation (2.67) implies that

$$A_{\mu\sqrt{n}}(e^{-\frac{s}{n}}) = \frac{\lambda_1(e^{-s/n})^{\mu\sqrt{n}} - \lambda_2(e^{-s/n})^{\mu\sqrt{n}}}{\lambda_1(e^{-s/n})^{\mu\sqrt{n}+1} - \lambda_2(e^{-s/n})^{\mu\sqrt{n}+1}}. \tag{2.101}$$

Finally, $A_{\mu\sqrt{n}}(e^{-\frac{s}{n}})$ can be expressed as

$$A_{\mu\sqrt{n}}(e^{-\frac{s}{n}}) = \frac{\exp(\mu\sqrt{n}\log\lambda_1(e^{-s/n})) - \exp(\mu\sqrt{n}\log\lambda_2(e^{-s/n}))}{\exp((\mu\sqrt{n}+1)\log\lambda_1(e^{-s/n})) - \exp((\mu\sqrt{n}+1)\log\lambda_2(e^{-s/n}))}. \quad (2.102)$$

Moreover, from (2.68) and (2.69) we have

$$B_{\mu\sqrt{n}}(e^{-\frac{s}{n}}) = A_{\mu\sqrt{n}}(e^{-\frac{s}{n}}), \quad (2.103)$$

$$C_{\mu\sqrt{n}}(e^{-\frac{s}{n}}) = A_{\mu\sqrt{n}}(e^{-\frac{s}{n}}) - A_{\mu\sqrt{n}-1}(e^{-\frac{s}{n}}). \quad (2.104)$$

Furthermore, from (2.79) we have

$$\begin{aligned} E[e^{-\frac{s}{n}T_1^{(\sqrt{n}\mu)}}] &= A_{\mu\sqrt{n}}(e^{-\frac{s}{n}}) + \frac{B_{\mu\sqrt{n}}(e^{-\frac{s}{n}})}{1-B_{\mu\sqrt{n}}(e^{-\frac{s}{n}})} \cdot C_{\mu\sqrt{n}}(e^{-\frac{s}{n}}) \\ &= A_{\mu\sqrt{n}}(e^{-\frac{s}{n}}) \frac{1-A_{\mu\sqrt{n}-1}(e^{-\frac{s}{n}})}{1-A_{\mu\sqrt{n}}(e^{-\frac{s}{n}})}, \end{aligned} \quad (2.105)$$

and $E[e^{-\frac{s}{n}T_1^{(\sqrt{n}\mu)}}]$ can be expressed as the following quotient

$$\frac{\lambda_1(e^{-s/n})^{\mu\sqrt{n}} - \lambda_2(e^{-s/n})^{\mu\sqrt{n}} - \frac{\lambda_1(e^{-s/n})^{\mu\sqrt{n}}}{\lambda_1(e^{-s/n})} + \frac{\lambda_2(e^{-s/n})^{\mu\sqrt{n}}}{\lambda_2(e^{-s/n})}}{\lambda_1(e^{-s/n})^{\mu\sqrt{n}}\lambda_1(e^{-s/n}) - \lambda_2(e^{-s/n})^{\mu\sqrt{n}}\lambda_2(e^{-s/n}) - \lambda_1(e^{-s/n})^{\mu\sqrt{n}} + \lambda_2(e^{-s/n})^{\mu\sqrt{n}}}. \quad (2.106)$$

If we consider enough terms to account for cancellations in the differences in (2.102) and (2.104) we obtain the expansion for $A_{\mu\sqrt{n}}(e^{-\frac{s}{n}})$ which coincides with $B_{\mu\sqrt{n}}(e^{-\frac{s}{n}})$, and we find also the expansion for $C_{\mu\sqrt{n}}(e^{-\frac{s}{n}})$ as $n \to \infty$.

Then, the asymptotic expansions (2.95), (2.97), (2.98), (2.96), (2.97) and (2.98) are combined with equations (2.99) and (2.100). Finally, a quotient of exponentials, which can be expressed as a hyperbolic tangent is obtained. □

The following limit results involve the hyperbolic function table, namely the hyperbolic tangent *tanh*, the hyperbolic cosecant *csch*, the hyperbolic secant *sech* and also the hyperbolic cotangent *coth*.

Proposition 2.6. *We have pointwise for $n \to \infty$ the following asymptotic relations*

$$E[e^{-\frac{\lambda}{n}T_1^{\mu\sqrt{n}}}] = 1 - \sqrt{2\lambda}\tanh(\mu\sqrt{2\lambda})\cdot n^{-\frac{1}{2}} + \mathscr{O}(n^{-1}), \tag{2.107}$$

$$E[e^{-\frac{\lambda}{n}T_1^{\mu\sqrt{n}}}; S(T_1^{\mu\sqrt{n}}) > 0] = 1 - \sqrt{2\lambda}\coth(\mu\sqrt{2\lambda})\cdot n^{-\frac{1}{2}} + \mathscr{O}(n^{-1}), \tag{2.108}$$

$$E[e^{-\frac{\lambda}{n}T_1^{\mu\sqrt{n}}}; S(T_1^{\mu\sqrt{n}}) \leq 0] = \sqrt{2\lambda}\,\mathrm{csch}(2\mu\sqrt{2\lambda})\cdot n^{-\frac{1}{2}} + \mathscr{O}(n^{-1}), \tag{2.109}$$

Proof. Elementary asymptotic calculations are performed, similarly as in the proof of Proposition 2.5. Clearly the entries (2.108) and (2.109), add up to (2.107) by the law of total probability. This corresponds in the limit to the elementary identity

$$\tanh\left(\frac{z}{2}\right) = \coth(z) - \mathrm{csch}(z), \tag{2.110}$$

which is equivalent to the identity (see [3, 4.5.30, P.84])

$$\tanh\left(\frac{z}{2}\right) = \frac{\cosh(z)-1}{\sinh(z)}. \tag{2.111}$$

□

Lemma 2.11. *Let*

$$h(x) = \frac{1}{\sqrt{2\pi}}\cdot x^{-3/2} + 2\sum_{k\geq 1}\left[\frac{1}{\sqrt{2\pi}}\cdot x^{-3/2} - 2\sqrt{\frac{2}{\pi}}k^2\mu^2\cdot x^{-5/2}\right]\exp\left(-\frac{2k^2\mu^2}{x}\right), \tag{2.112}$$

then, the following holds

$$\int_0^\infty (1-e^{-\lambda x})h(x)dx = \sqrt{2\lambda}\tanh\left(\mu\sqrt{2\lambda}\right). \tag{2.113}$$

Proof. After the termwise integration, the resulting series are summed. □

Theorem 2.3. *The Laplace transform of the scaled full avalanche length for the simple symmetric random walk satisfies:*

$$\lim_{n\to\infty} E[e^{-\frac{\lambda}{n}L^*_{\mu\sqrt{n},\varepsilon n}}] = \frac{\int_\varepsilon^\infty h(x)dx}{\int_0^\varepsilon (1-e^{-\lambda x})h(x)dx + \int_\varepsilon^\infty h(x)dx}. \tag{2.114}$$

Proof. The limit results for the first trade in Proposition 2.5, Lemma 2.11 and Lemma 5.1 imply this theorem. □

Remark 2.13. *Note that by integrating the series for h term by term, series representations for the integrals in the numerator and denominator in (2.114) are found. The terms for* $\int_0^\varepsilon (1-e^{-\lambda x})h(x)dx$ *can be expressed in terms of the error and complementary error functions. Termwise integration of* $\int_\varepsilon^\infty h(x)dx$ *yields that there exists a series representation which can be recognize as a series representation of a Jacobi theta function. We have*

$$\lim_{n\to\infty} P\left[\frac{1}{n}T_1^{(\mu\sqrt{n})} > \varepsilon\right] = \vartheta_4(0,e). \tag{2.115}$$

2.5 An initially empty order book

So far, the analysis is performed with an assumption (2.1), i.e. with an initially full order book. Let us now assume that an order book is initially empty, i.e.

$$V(0,u) = 0, \quad u \in \mathbb{Z}. \tag{2.116}$$

Furthermore, as in the case when the initially full limit order book is considered, the dynamics evolves with respect to the equation (2.2). In order to be precise, we specifically define trading times $\{\widetilde{\tau}_i : i \geq 0\}$ and intertrading times $(\widetilde{T}_i)_{i\geq 1}$ that correspond to the model when the limit order book is initially empty by

$$\widetilde{\tau}_0 = 0, \quad \widetilde{\tau}_i = \inf\{n > \widetilde{\tau}_{i-1} : V(n,S_n) > 0\}, \quad \widetilde{T}_i = \widetilde{\tau}_i - \widetilde{\tau}_{i-1}, \quad i \geq 1. \tag{2.117}$$

Note that a trade event occurs at time n, if $V(n,S_n) > 0$. Following dynamics (2.2), orders get placed with a fixed displacement μ from the mid-price and get executed when the mid-price reaches their level. Therefore, the first trade (either a Type I or a Type II trade) in an initially empty limit order book occurs at the smallest time $\widetilde{T}_1$ such that

$$|\min(S(0),S(1),\ldots,S(\widetilde{T}_1-1)) - S(\widetilde{T}_1)| = \mu.$$

Remark 2.14. *Due to the fact that the limit order book model dynamics follow (2.2), note that if we start with an initially empty order book after the first trade occurs (either a Type I or a Type II trade) the analysis of the avalanche length is equivalent to what we had with an initially full order book.*

2.5.1 The first Type I trade in an initially empty book

Before the first Type I trade in an initially empty book occurs at the price level k, $k \in \{1,2,...,\mu\}$, mid-price needs to go down to the price level $k-\mu$ and the order gets placed with a fixed displacement μ from the mid price. Therefore, for $k \in \{1,2,...,\mu\}$ the path $(S(0),S(1),...,S(\widetilde{T}_1))$ goes down by $\mu-k$ steps, but not below, and then goes up to the level k, as illustrated in Figure 2.13. Specifically, Figure 2.12 depicts the trivial case when $k=\mu$.

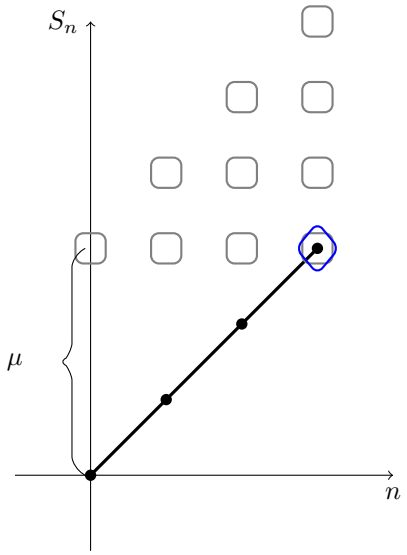

Figure 2.12: *Example of the path that depicts the trivial case for the first Type I trade in an initially empty order book: a new order is placed at the starting time at distance* $S_0+\mu=\mu$ *and the first Type I trade occurs at the level* μ*, after exactly* μ *steps up.*

For $n \geq 1$, $\mu \geq 1$ and $k \in \{1,2,...,\mu\}$ we introduce the paths:

$$\mathscr{G}_{n,\mu,k} = \left\{ s \in U_n : s_0 = 0,\, k-\mu < s_j < \mu \text{ for } 1 \leq j \leq n-1 \text{ and } s_n = k-\mu \right\}, \quad (2.118)$$

$$\mathscr{F}_{n,\mu,k} = \left\{ s \in U_n : s_0 = k-\mu,\, k-\mu-1 < s_j < k \text{ for } 1 \leq j \leq n-1 \text{ and } s_n = k \right\}. \quad (2.119)$$

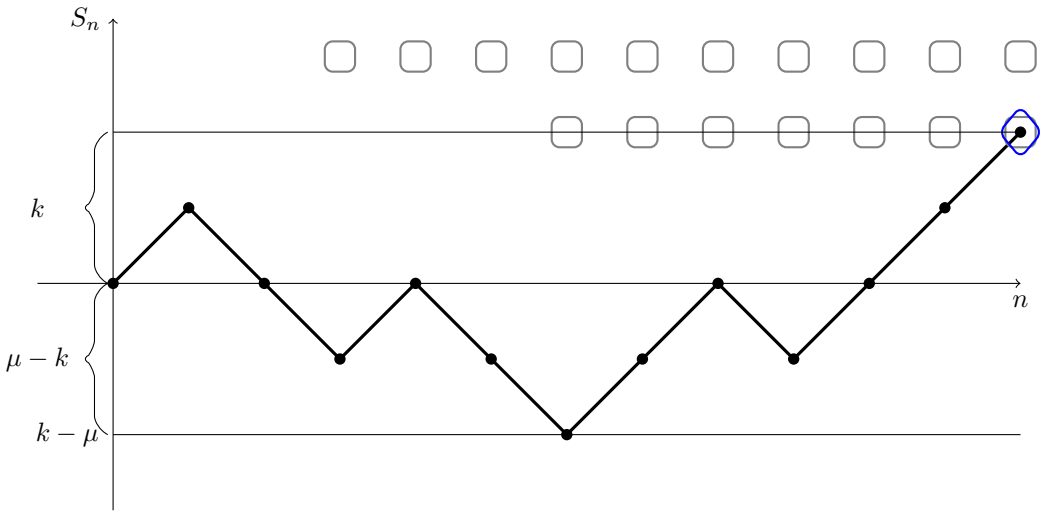

Figure 2.13: *The path of the first Type I trade in an initially empty order book can be seen as a concatenation of two paths: first one which is starting at 0, forbidden level is μ and absorbing at level $k-\mu$; second path which is starting at $k-\mu$, forbidden level is $k-\mu-1$ and absorbing at level k*

.

Furthermore, we define the corresponding sets with arbitrary, finite length paths as:

$$\mathscr{G}_{\mu,k} = \bigcup_{n\geq 1} \mathscr{G}_{n,\mu,k}, \quad \mathscr{F}_{\mu,k} = \bigcup_{n\geq 1} \mathscr{F}_{n,\mu,k} \tag{2.120}$$

As introduced before, $\lambda_1(s)$ and $\lambda_2(s)$ are defined as:

$$\lambda_1(s) = \frac{1+\sqrt{1-s^2}}{s}, \quad \lambda_2(s) = \frac{1-\sqrt{1-s^2}}{s}. \tag{2.121}$$

Lemma 2.12. *For $k \in \{1,2,...,\mu\}$ denote by $G_{\mu,k}, F_{\mu,k}$ the probability generating functions for the combinatorial classes $\mathscr{G}_{\mu,k}, \mathscr{F}_{\mu,k}$ respectively. It holds that*

$$G_{\mu,k}(s) = \frac{\lambda_1(s)^\mu - \lambda_2(s)^\mu}{\lambda_1(s)^{2\mu-k} - \lambda_2(s)^{2\mu-k}}, \tag{2.122}$$

$$F_{\mu,k}(s) = \frac{\lambda_1(s) - \lambda_2(s)}{\lambda_1(s)^{\mu+1} - \lambda_2(s)^{\mu+1}}. \tag{2.123}$$

Proof. The generating function $G_{\mu,k}$ is derived following [51, XIV.4, (4.11), P.351], and by shifting a path from $\mathscr{G}_{n,\mu,k}$ by $\mu-k$. The shifted path can be viewed as a path cor-

responding to absorption at zero at the n-th trial in a symmetric game of gamblers ruin with initial position $\mu - k$ and absorbing barriers at 0 and $2\mu - k$. The formula [51, XIV.4, (4.11), P.351] for $p = 1/2$, $q = 1/2$, $z = \mu - k$, $a = 2\mu - k$ yields (2.122).

The generating function F_μ is derived following[51, XIV.4, (4.12), P.351] and by shifting a path from $\mathscr{F}_{n,\mu,k}$ by $\mu - k + 1$. Therefore, the formula (2.123) is obtained by plugging values $p = 1/2$, $q = 1/2$, $z = 1$, $a = \mu + 1$ in the formula [51, XIV.4, (4.12), P.351]. □

Since the generating function $F_{\mu,k}$ does not depend on k, from now on we can omit subscript k and instead of $F_{\mu,k}$ write F_μ.

2.5.2 The first Type II trade in an initially empty book

The path of the first Type II trade in the initially empty order book needs to go down by μ or more, and then to go up by μ steps. Therefore, if the first trade in the initially empty order book is a Type II trade, then the path $(S(0),\dots,S(\widetilde{T}_1))$ is a concatenation of $K \geq 1$ elements from $\mathscr{B}_\mu$ and one element from $\mathscr{C}_\mu$. The proof is same as in the Proposition 2.3.

2.5.3 The first trade in an initially empty book

Proposition 2.7. *The generating function of the time to the first trade when the order book is initially empty is given by*

$$E[z^{\widetilde{T}_1}] = \sum_{k=o}^{\mu} G_{\mu-k}(z)F_\mu(z) + \frac{B_\mu(z)}{1-B_\mu(z)} \cdot C_\mu(z). \tag{2.124}$$

Proof. Let us define the set $\widetilde{\mathscr{T}_{\mu,n}}$ of all paths where the first trade in an initially empty order book occurs at step n. and $\widetilde{\mathscr{T}_\mu} = \bigcup_{n\geq 1} \widetilde{\mathscr{T}_{\mu,n}}$. This set contains elements that are either Type I trades or Type II trades. By Lemma 2.12 the path of the Type I trade is the concatenation of an element from the class $\mathscr{G}_{\mu,k}$ and an element from the class $\mathscr{F}_{\mu,k}$. The path of the Type II trade corresponds bijectivly to the Cartesian product of the set of finite sequences of length 1 or more of elements in $\mathscr{B}_\mu$ times the class $\mathscr{C}_\mu$, see Proposition 2.3. Using the symbolic notation from [53, Section I] this is written more clearly as

$$\widetilde{\mathscr{T}_\mu} = \sum_{k=o}^{\mu} \mathscr{G}_{\mu-k}(z)\mathscr{F}_\mu(z) \cup \mathrm{SEQ}_{\geq 1}(\mathscr{B}_\mu) \times \mathscr{C}_\mu. \tag{2.125}$$

From [53, Theorem I.1, P.27] and the section on *restricted constructions*, in particular $\text{SEQ}_{\geq k}$, in [53, P.30], the probability generating function (2.124) is obtained. □

2.6 On a binomial limit order book model in which the mid-price converges in distribution to a subordinated Wiener process

The setup of the model which is introduced in this section, is motivated by the theory presented in the papers [84], [83] and [85]. Some inspiration from the aforementioned work is presented. As introduced in [84], a continuous time random walk (CTRW) is a random walk defined by a sequence of independent, identically distributed random vectors $(X_i, Y_i)_{i\geq 0}$ such that X_i represents the i-th particle jump, while $Y_i > 0$ represents the waiting time preceding to the i-th jump. In [85] the authors consider the model in which the n-th particle jump of the random walk is preceded by a waiting time W_n, such that $P(W_n > t) \approx t^{-\beta}$, $0 \leq \beta \leq 1$. Further, the authors observed that this process converges to a Brownian motion, with a slower spreading rate $t^{\frac{\beta}{2}}$ than the $t^{\frac{1}{2}}$, which is the usual Brownian motion spreading rate. The properties of the inverse stable subordinator are comprehensively reviewed in [83].

2.6.1 The model

We introduce a stochastic model of a limit order book, driven by a simple symmetric random walk such that each step is preceded by a waiting time.

Let $(\Omega, \mathscr{F}, P)$ be a probability space, on which the following two independent processes X and Y are defined as follows.

- Define $X = (X_n)_{n\geq 0}$ as the sequence of independent and identically distributed (i.i.d.) random variables such that $E[X_n] = 0$ and $\text{Var}[X_n] = 1$, $n \geq 0$. More precisely, let $X = (X_n)_{n\geq 0}$ be a simple symmetric random walk.
- Let $Y = (Y_n)_{n\geq 0}$ be the sequence of independent and identically distributed (i.i.d.) random variables, such that the following convergence in distribution holds

$$\frac{\sum_{k=1}^{[nt]} Y_k}{n^{-\alpha}} \Rightarrow U_\alpha(t), \quad n \to \infty,$$

where $\{U_\alpha(t)\}$ is α-stable subordinator.

For a discussion on convergence to stable laws [122, Chapter 4, P.114]. Further, in [62] authors discussed the three models of subordinated processes and their characteristics.

Denote by $S(n) = \sum_{k=1}^{n} X_k$ the particle location after n jumps and let $T(n) = \sum_{k=1}^{n} Y_k$ be the time when the n-th jump occurs.

Furthermore, we define the time-inverse process $(A(n))_{n\geq 0}$ of the process $(T(n))_{n\geq 0}$ as

$$A(n) = \inf\{k \geq 0 : T(k) > n\}, \quad n \geq 0.$$

Now, define the order compound price process $Z = (Z(n))_{n\geq 0}$ as

$$Z(0) = 0, \quad Z(n) = S(A(n)). \tag{2.126}$$

This process $Z = (Z(n))_{n\geq 0}$ is interpreted as a mid-price process.

From now on, the focus is only on the ask side, since by symmetry all calculations can be established also for the bid side. Intuitively, a trade on the ask-side occurs when the mid-price moves up. With the following definition trading times on the ask side are defined.

Definition 2.6. Define the trading times $\{\rho_i : i \geq 0\}$ and the intertrading times $(R_i)_{i\geq 1}$ as

$$\rho_0 = 0, \quad \rho_i = \inf\{n > \rho_{i-1} : Z_n > Z(\rho_{i-1})\}, \quad R_i = \rho_i - \rho_{i-1}, \quad i \geq 1. \tag{2.127}$$

2.6.2 The convergence to the subordinated Brownian motion

It follows from Donsker's theorem that

$$\left\{\frac{1}{\sqrt{n}} S_{\lfloor nt \rfloor} : t \geq 0\right\} \to \{W(t) : t \geq 0\} \tag{2.128}$$

in distribution on the Skorohod space, where $\{W(t) : t \geq 0\}$is a Wiener process, see [14, Theorem 14.1. of XIV,P.146].

We also have $\frac{N(n^\alpha t)}{n} \Rightarrow U_\alpha(t)$, because of the assumptions from the beginning.

The next theorem is Theorem 6.7 in [14, P.70].

Theorem 2.4. *(Skorokhod theorem on a joint probability space) Suppose that $P_n \Rightarrow_n P$ and P has a separable support. Then there exist random elements ξ_k and ξ , defined*

on a common probability space $(\Omega, \mathscr{F}, P)$, *such that* $\mathscr{L}(\xi_n) = P_n$ *and* $\mathscr{L}(\xi) = P$ *and* $\xi_n(\omega) \to_n \xi(\omega)$ *for every* ω.

Apply the previous theorem and select copies $\widetilde{S_n}$ and $\widetilde{N_n}$ such that almost surely

$$\frac{\widetilde{S}([nt])}{\sqrt{n}} \to \widetilde{W}(t), \quad n \to \infty, \text{ in } C([0,\infty])$$

$$\frac{\widetilde{T}([n^\alpha t])}{n} \to \widetilde{U}_\alpha(t), \quad n \to \infty, \text{ in } D([0,\infty]),$$

where $D([0,\infty])$ is a Skorokhod space.

In the proof of the next theorem we refer to three auxiliary lemmas that are presented in Appendix 5.4.

Theorem 2.5. *The scaled mid-price process converge in distribution to a subordinated Wiener on* $D([0,\infty])$

$$\frac{\widetilde{S_n}(\widetilde{A_n}(n^{-\alpha}t))}{\sqrt{n}} \to \widetilde{W}\left(\widetilde{U}_\alpha^{(-1)}(t)\right).$$

Proof. By Lemma 5.2, we are in position to conclude

$$\frac{\widetilde{S_n}(\widetilde{T_n}(n^\alpha t))}{\sqrt{n}} = \frac{\widetilde{S_n}\left(n\frac{\widetilde{T_n}(n^\alpha t)}{n}\right)}{\sqrt{n}} \to \widetilde{W}\left(\widetilde{U}_\alpha(t)\right) \quad n \to \infty, \text{ in } D([0,\infty]).$$

Therefore, since

$$\frac{\widetilde{S_n}(\widetilde{T_n}(n^\alpha t))}{\sqrt{n}} \stackrel{d}{=} \frac{S(T(n^\alpha t))}{\sqrt{n}}$$

we have convergence in distribution

$$\frac{S(T(n^\alpha t))}{\sqrt{n}} \Rightarrow W\left(U_\alpha(t)\right) \quad n \to \infty, \text{ in } D([0,\infty]).$$

Applying Lemma 5.3, Lemma 5.4 and Skorokhod's theorem, almost sure convergence for copies is obtained, i.e. for $n \to \infty$, in $D([0,\infty])$ we have:

$$\frac{\widetilde{S_n}(\widetilde{A_n}(n^{-\alpha}t))}{\sqrt{n}} = \frac{\widetilde{S_n}\left(n\frac{\widetilde{A_n}(n^{-\alpha}t)}{n}\right)}{\sqrt{n}} = \frac{\widetilde{S_n}\left(n\left(\frac{\widetilde{T_n}(nt)}{n^{-\alpha}}\right)^{(-1)}\right)}{\sqrt{n}} \to \widetilde{W}\left(\widetilde{U}_\alpha^{(-1)}(t)\right).$$

Hence, in our model settings a subordinated Wiener process is achieved in a limit of the rescaled mid-price process. □

2.6.3 On the paths of Bernoulli excursion process

We introduce the paths of Bernoulli excursion process in order to describe a path that leads to the trade event.

Let $\mathscr{W}^-_{N,0}$ be the set of paths of length $N \geq 0$ that go from 0 to 0 and never enter the strictly positive numbers, i.e.

$$\mathscr{W}^-_{N,0} = \{(S(0), S(1), ..., S(N)) : S(0) = 0, S(N) = 0, S(i) \leq 0, i = 1, ..., N-1\}.$$

Furthermore, define by $\mathscr{W}^-_N$ the set of paths of length $N \geq 0$ that go from 0 to 0 and never hit 0 in between but also that never enter the strictly positive number:

$$\mathscr{W}^-_N = \{(S(0), S(1), ..., S(N)) : S(0) = 0, S(N) = 0, S(i) < 0, i = 1, ..., N-1\}.$$

Next the corresponding sets with arbitrary, finite path length, are defined as

$$\mathscr{W}^-_0 = \bigcup_{N \geq 1} \mathscr{W}^-_{N,0}, \quad \mathscr{W}^- = \bigcup_{N \geq 1} \mathscr{W}^-_N. \tag{2.129}$$

Let W^-_0 and W^- denote the counting generating functions for the number of steps corresponding to the paths $\mathscr{W}^-_0$ and $\mathscr{W}^-$ respectively.

Note that any path from $\mathscr{W}^-_0$ can be decomposed as a sequence of the paths from $\mathscr{W}^-$, and each path from $\mathscr{W}^-$ can be decomposed as a Cartesian product of one step down, one or few paths from $\mathscr{W}^-_0$, and one step up.

$$\mathscr{W}^-_0 = \mathrm{SEQ}(\mathscr{W}^-), \qquad \mathscr{W}^- = \left\{ \diagdown \right\} \times \mathscr{W}^-_0 \times \left\{ \diagup \right\}. \tag{2.130}$$

Therefore, following [54, P.27], we have:

$$W^-_0(s) = \frac{1}{1 - W^-(s)}, \qquad W^-(s) = s^2 W^-_0(s). \tag{2.131}$$

Combining those two equations, we have:

$$W^-_0(s) = \frac{1 - \sqrt{1 - 4s^2}}{2s^2}. \tag{2.132}$$

By Definition 2.6, we say that a trade event occurs whenever the price maximum

increases. If we denote by $\mathscr{T}$ the set of paths for trading excursions that lead to the trade, we have:

$$\mathscr{T} = \mathscr{W}_0^- \times \left\{ \right\}. \tag{2.133}$$

Denote by $T(s)$ the corresponding counting generating function for the number of steps corresponding to the paths $\mathscr{T}$, following [53, P.27], we have $T(s) = sW^-(s)$, i.e.:

$$T(s) = \frac{1-\sqrt{1-4s^2}}{2s}. \tag{2.134}$$

Let a_k the number of paths from the set $\mathscr{T}$ of length k, then by expanding (2.134) with the binomial theorem we have:

$$a_k = \frac{1}{k}\binom{k}{(k+1)/2} \tag{2.135}$$

when k is odd, and $a_k = 0$ otherwise.

Furthermore, this model can be extended after introducing the order book dynamics as in the equation (2.2). Therefore, this model can be further studied. The possible point of interest can be to try to explain avalanche length in this model settings and to study its distribution, as it was done in subsection 2.4.2.

Example of a simple discrete distribution with positive α-stable limit Let us define a distribution of the random variable Y_k as

$$P(Y_k = n) = (-1)^{n-1}\binom{\alpha}{n}, \quad n \geq 1$$

where $0 < \alpha < 1$.

Further, the Binomial Theorem yields that the probability generating function $m(z)$ of the random variable Y_k is given by

$$m(z) = 1 - (1-z)^{\alpha}.$$

Then, we have

$$\lim_{n\to\infty} m(-\frac{z}{n^{1/\alpha}})^n = \exp(-z^{\alpha}),$$

which is the Laplace transform of a positive α-stable subordinator.

3

Machine learning in finance

This Chapter is devoted to applying machine learning techniques on the limit order book market data. It contains enhanced versions of the papers [100], [99] and [98], while some theoretical concepts are also presented in [97]. Note that these papers were produced during the realization of research that is part of this PhD thesis. This study is observing the informativeness of features extracted from the limit order book data plugged into the Gated Recurrent Unit (GRU) network model to classify market data vectors into the label (buy/idle). Further, the performance of the GRU network model fed with the features selected with respect to the proposed methods is compared with the model that has as input randomly chosen features.

3.1 Motivation

The past years have brought a new era of automation that has affected a wide range of fields that have an impact on human lives. As we have entered an era of big data (see [88]), machine learning has become present in various fields and there is a possibility of its application in trading. Since trading has experienced automatization and digitization, stocks have substituted the physical trading concept by electronic trading. While an average human's eye blinks last up to 400 milliseconds, the high-frequency trading (HFT) algorithms are processing information with respect to the time-scale of the microseconds. Therefore, more trades are occurring automatically and the whole role of a trader is going into the direction of a data scientist, who uses algorithms based on Machine Learning for automatic trading (see [78]).

The traditional stock exchange, which was physically located, is substituted by electronic stock exchanges. For a detailed explanation of an electronic marketplace mechanism see [22], while a comprehensive overview of practical aspects and rules of market microstructure trading is given in [63].

There is a potential in analyzing historical stock data as well as in developing algorithms to analyze that data, since financial markets are informative and convey memory properties according to [127], [32] and [92].

3.2 Related literature

This section is devoted to a brief review of the recent literature related to the application in finance of numerous approaches, mainly based on different artificial intelligence techniques. A broad range of research has been performed in the direction of analyzing the financial data and constructing adequate mathematical models to describe and forecast the stock market behavior.

Forecasting future trends of the stock market using historical data has become very attractive for both researchers from academia and business, see for example [39]. In [39] the researchers explored the feature optimization capacity of firefly with an evolutionary framework and the performance of the proposed model was evaluated using different stock market datasets. Moreover, various techniques for forecasting of stock markets are provided in [91], including an approach based on rule generation by rough set theory and fuzzy sticks data discretization. For option price forecasting, the nonlinearity and non-stationarity of option dynamics are the main challenges, so the authors in [70] developed an improved version of the adaptive filter for online option trading with good performance. The authors in [94] compared four prediction models, Artificial Neural Network (ANN), Support Vector Machine (SVM), random forest and naive-Bayes with two approaches for input to these models, with the aim to predict the direction of movement of stock and stock price index for Indian stock markets.

On the other hand, a stock market trading system based on fuzzy logic rules was proposed in [20] and both direct and indirect channels of foreign information transmission were investigated. A stock-based collaborative filtering algorithm for stock prediction proposed in [128] relies on the transmission effect among different stocks. A challenging topic of the impact of investor sentiment on stock market volatility was explored in [123],

where the authors proposed a new model and implemented nonlinear quantile regression on mixed frequency data by introducing a kernel function. This approach was shown to be superior to the previously used ones for the same problem, according to empirical results. Furthermore, a novel fuzzy recommendation system for stock market investors was proposed in [89] and it was shown that this approach is profitable and presents high stability.

The leverage of Machine Learning, as well as Artificial Intelligence in general, has attracted the attention of academic researchers in many fields, see [80]. This topic is interesting for hedge funds and financial institutions, but also for scientists, who have obtained several high-impact applications of it in finance. Various Artificial Intelligence approaches have been used for, usually very challenging, financial forecasting, which has been attracting many researchers.

The authors in [108] presented a universal model for different stocks to predict the direction of price moves using Deep Learning Neural Networks (DLNNs) given the price and order flow history. An approach for hedging a portfolio of derivatives using reinforcement learning methods, which outperforms the hedging in a Heston model without transaction costs, is proposed in [21]. Additionally, the authors in [26] used deep learning and boosting approaches in order to predict stock market crisis episodes. They presented an interesting forecasting mechanism of the probability of a stock market crash event in various time frames, combining different machine learning algorithms. Moreover, financial network indicators that can be applied to global stock market investment strategies were investigated in [79] and combined with several machine learning approaches. The first 3-dimensional convolutional neural networks (CNN) model designed for stock market prediction was proposed in [68] and the presented framework significantly outperformed shallow artificial neural networks. The authors in [96] concluded that the random forest algorithm model had the best classification effect when compared with traditional statistical methods and it was found that credit and personal information risk were the most important factors in the future development of Internet finance when back propagation neural network was used to evaluate these risks. On the other hand, a weighted support vector machine (WSVM) method was used in [27] to predict stock trading signals.

In [57] seven different machine and deep learning algorithms were proposed to learn the inherent patterns and to predict future movements in the stock market. Since, there are various factors that have an influence on the stock market, using adequate technical indicators is important in achieving good forecasting results, see for example [129]. The

authors in [91] provide a review on various techniques that perform well in the forecasting of stock markets. Additionally, they propose an approach through data discretization by fuzzistics and rule generation by rough set theory. An application of the deep learning architectures for a classification modeling in high-frequency trading used to predict short-term price movement based on the information of the market depth is developed in [46]. Furthermore, the next price-flip event in limit order book using the ability of the RNN is modeled in [45].

The capability of online one-pass algorithms for solving issues in HFT is established in [82]. On the other hand, the novel concept to abstract the knowledge base (KB) component from the fuzzy rule based system (FRBS) in order to predict the stock market is obtained in [7]. In [107] authors observe the influence of the Window Size on the performance of the models used to predict the future price direction and technical indicators were used as input.

Exploring the text data and the influence of social networks in finance is interesting for scientists: a treatment of the influence of the social network on a stock market is established in [16] and [125], while in [103] a method extracting relevant information about financial risks from descriptive text data using a deep learning approach is presented. In [24] authors have evaluated systemic risk using the market data enhanced with the text data from financial tweets. The authors in [118] improved the time-varying risk-adjusted performance of trading systems of AI models. Since there are many choices which price can be considered (such as mid-price, best ask price, last, previously traded price, etc.) in [41] authors introduced a notion of the efficient price and a methodology to statistically estimate this price.

Note that the research included in this thesis is not devoted to foresee future market performance with certainty. Rather, the interest is put on the potential of the neural network model to identify profitable trades, i.e. to map each vector of market data into a label (buy/idle).

3.3 The technology used to support data processing

In order to deal with an immense amount of data such as stock market data, the utilization of adequate technology and tools is as crucial as choosing effective techniques to handle that data.

Apache Spark: When it comes to choosing a good system for large-scale data processing, Apache Spark[1] is a general-purpose analytic engine especially designed for that.

From the programmers' point of view, one of the main advantages worth emphasizing is that it has the property of easy parallelism implementation. Since, parallelization is beneficial for large-scale data processing, having this part almost automatically employed, allows programmers to completely focus on the main logic of the tasks needed. As it turns out this parallelization property is successfully employed during the data transformation part (described in Section 3.7).

One more aspect of this technology that can be very useful application-wise is that it is possible to expand hardware when needed due to horizontal scaling. Moreover, there are many different application programming interfaces (APIs) available for multiple programming languages: Java, Scala, Python, R.

In order to enable working with the large-scale data set, a general-purpose processing approach is proposed to extract the final storage data from the raw market data by applying carefully chosen transformations (described in Section 3.7). After that, the final storage data can be accessed with the available APIs that allow the operative processing of the financial data.

MongoDB: In order to provide a place to store such a large amount of data in an efficient way, MongoDB[2] was utilized. MongoDB is a cross-platform general-purpose, document-based, distributed, NoSQL database built for modern application development. Among the main reasons for choosing this specific database, data in MongoDB is stored in JSON-like documents, which makes the database very flexible and scalable.

Note that in this research, programming language Python was chosen due to its convenience in working with data and broad usability. The architecture of the data processing pipeline employed in this research is depicted in Figure 3.1, which also depicts the proposed stages for the data transformation (Raw aggregation, Technical analysis, Data lapsing, Data labeling) that are explained later in Section 3.7.

[1] Apache spark https://spark.apache.org/
[2] https://www.mongodb.com/

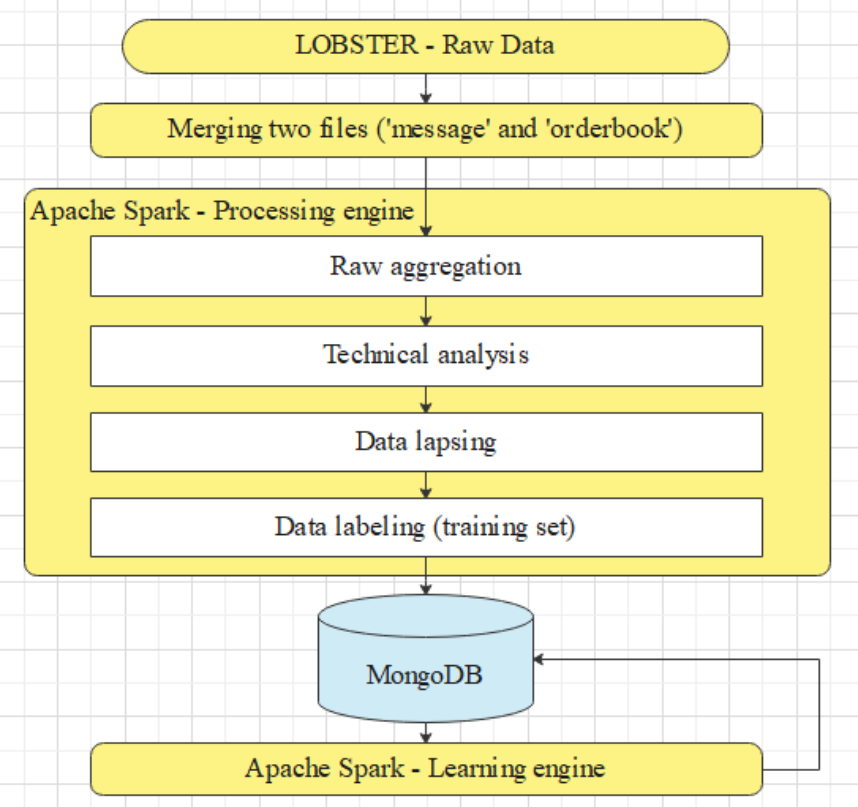

Figure 3.1: *The general architecture of the processing pipeline.*

The aforementioned data processing platform enables the further application of cutting-edge algorithms and machine learning techniques. Specifically, this platform is compatible with the LOBSTER data, and therefore it enables us to extract features from the limit order book and to employ data transformation in order to prepare data into a format suitable for research. The observation is made on the performance of processing AAPL (Apple stock) data of 32 consecutive trading days starting from the 1st of May, 2015. The performance of the simple sequential implementation is compared with respect to the implementation running on top of the Apache Spark based platform.

3.3.1 Performance evaluation

Here the main focus is acquiring an approximated factor that measures the speed-up obtained by the incorporation of the Apache Spark. With the platform being constructed, the measurements were run in order to compare the processing time required to extract

the same set of features for a single-threaded, sequential, ad-hoc implementation (benchmark) and the proposed implementation running on top of the Apache Spark engine. Two measurements are performed on the data including 32 trading days starting from the 1st of May, 2015. Regarding the set of features, for both implementations, the *TA-Lib* technical indicators are used and the computational complexity of all the technical indicators is $O(n)$. The results of the measurements are established in Table 3.1 (as presented in Table 1 in [99]). Note that the basic sequential implementation is called benchmark and it is used for the comparison with the proposed engine.

Measurement type	Average duration of processing single trading day [minutes]	Std. dev. of processing single trading day [minutes]	Total duration for 32 trading days [minutes]
The benchmark	136	38	4352
The proposed engine	24.5	4.1	784

Table 3.1: *Performance evaluation of the processing time required for 32 trading days*

To conclude, the proposed Appache Spark engine overall provided the speed up of 5.55 times when compared to the implemented sequential version. Note that the test was run on eight CPU cores, but the improvement is significant.

3.4 The data used in this research

The NASDAQ, formerly known as an acronym for the National Association of Securities Dealers Automated Quotation, is the second-largest stock market in the world (measured by market capitalization of shares traded). It was founded in 1971 in New York City. Many of the world-leading companies are listed on NASDAQ, such as Facebook, Tesla, Amazon, Apple, Microsoft, Alphabet, Cisco, etc. For this research LOBSTER[3] (Limit Order Book System-Efficient Reconstructor) online limit order data is used. More precisely, LOBSTER is a limit order book tool that replicates the entire NASDAQ stock exchange order book by utilizing the NASDAQ's Historical TotalView-ITCH files. The data is available for each ticker (e.g. Tesla (TSLA), Apple (AAPL), Microsoft (MSFT))

[3]Lobster academic research data. https://lobsterdata.com. Accessed: 2020-08-05.

from the 27th of June 2007 up to the day before yesterday. LOBSTER records all submissions, executions of limit orders, cancellations, executions of hidden order, deletion, etc. For any selected trading day LOBSTER outputs two files, namely 'orderbook' file and 'message' file. The 'orderbook' file keeps track of all quotes (prices) and the corresponding depths (volume) for each timestamp up to the requested number of price levels. Note that for any selected trading day corresponding 'orderbook' and 'message' files have the same number of rows, which is the number of events that occurred on the particular selected trading day. The 'message' file consists of the following columns:

- *Time* represents the timestamp (measured in milliseconds after mid-night, 9:30pm is represented as 9.5*60*60=34200).
- *Type* represents the type of event that causes the update: 1 denotes limit order submission, 2 cancellation, 3 deletion of an order, 4 visible limit order execution, 5 hidden limit order execution, 6 indicates a cross trade, 7 trading halt. Note that a cross trade (e.g. auction trade), indicated by the type event 6, is not a trade. A cross trade is interpreted as an aggregated characterization of limit and hidden order executions.
- *Order ID* is a uniquely identification number assigned by the exchange.
- *Size of the order* is order size for the limit order submission, traded/canceled volume for the order execution/cancellation.
- *Price* is the price of the limit order (dollar price times 10000).
- *Direction of a trade*: -1 stands for sell limit order, while +1 stands for buy.

For a sample of the 'orderbook' file see Table 3.2 and for a sample of the 'message' file see Table 3.3.

3.5 The main idea

Note that a statistical arbitrage strategy (*Stat Arb* or *StatArb*) is a group of short-term trading strategies, based on the mathematical modeling techniques. The main job of the strategy is to identify inefficiency between securities in order to recognize a profitable

Table 3.2: *The 'orderbook' file sample of 3 price level LOBSTER data for TSLA ticker (Tesla, Inc.) company on January 7th 2019.*

Level 1	Level 1	Level 1	Level 1	Level 2	Level 2	Level 2	Level 2	Level 3	Level 3	Level 3	Level 3
Ask price	Ask size	Bid price	Bid size	Ask price	Ask size	Bid price	Bid size	Ask price	Ask size	Bid price	Bid size
3217200	24	3213200	5	3219000	100	3211900	10	3220000	2120	3211000	3
3219000	100	3213200	5	3220000	2120	3211900	10	3223200	15	3211000	3
3219000	100	3217100	100	3220000	2120	3213200	5	3223200	15	3211900	10
3219000	100	3217100	100	3220000	2120	3214000	100	3223200	15	3213200	5
3219000	100	3217100	100	3220000	2120	3214000	200	3223200	15	3213200	5
3219000	100	3217100	100	3220000	2120	3214000	100	3223200	15	3213200	5
3219000	100	3217100	100	3220000	2120	3213200	5	3223200	15	3211900	10
3219000	100	3213200	5	3220000	2120	3211900	10	3223200	15	3211000	3
3215000	49	3213200	5	3219000	100	3211900	10	3220000	2120	3211000	3
3215000	49	3213200	2	3219000	100	3211900	10	3220000	2120	3211000	3
3215000	49	3213200	2	3219000	100	3212500	5	3220000	2120	3211900	10
3219000	100	3213200	2	3220000	2120	3212500	5	3223200	15	3211900	10
3219000	49	3213200	2	3220000	2120	3212500	5	3223200	15	3211900	10
3219000	49	3215000	100	3220000	2120	3213200	2	3223200	15	3212500	5
3219000	49	3213200	2	3220000	2120	3212500	5	3223200	15	3211900	10
3218200	100	3213200	2	3219000	49	3212500	5	3220000	2120	3211900	10
3219000	49	3213200	2	3220000	2120	3212500	5	3223200	15	3211900	10
3219000	49	3213200	2	3220000	2120	3212500	5	3223200	15	3211900	10
3219000	49	3215400	100	3220000	2120	3213200	2	3223200	15	3212500	5
3216600	2	3215400	100	3219000	49	3213200	2	3220000	2120	3212500	5
3216600	2	3215400	101	3219000	49	3213200	2	3220000	2120	3212500	5
3216600	2	3215400	101	3217000	2	3213200	2	3219000	49	3212500	5
3216600	2	3215400	101	3217000	2	3215100	1	3219000	49	3213200	2

Table 3.3: *The 'message' file sample of the LOBSTER data for TSLA ticker (Tesla, Inc.) company on January 7th 2019. The sample presents data of order events for the time period less than a five second starting from 9:30 pm. Note that time is measured in milliseconds after the midnight.*

Time	Type	Order ID	Size	Price	Trade Direction
34200.18	4	10589488	2	3217100	-1
34200.18	4	9986208	24	3217200	-1
34200.18	1	10900144	100	3217100	1
34200.18	1	10900160	100	3214000	1
34200.18	1	10900176	100	3214000	1
34200.18	3	10900160	100	3214000	1
34200.18	3	10900176	100	3214000	1
34200.18	3	10900144	100	3217100	1
34200.19	1	10902168	49	3215000	-1
34200.42	4	9902468	3	3213200	1
34200.47	1	10946812	5	3212500	1
34200.56	4	10902168	49	3215000	-1
34200.56	4	9037520	51	3219000	-1
34200.56	1	10961024	100	3215000	1
34200.56	4	10961024	100	3215000	1
34200.56	1	10961088	100	3218200	-1
34200.56	3	10961088	100	3218200	-1
34200.79	1	10097796	100	3215400	1
34200.79	1	7964640	2	3216600	-1
34200.79	1	10309868	1	3215400	1
34200.79	1	9227064	2	3217000	-1
34200.79	1	9816456	1	3215100	1

situation. The overview of the trading strategy development and strategy evaluation is described in [93].

Let x_t be defined by the market vector data at time t that depicts the LOB shape at time t. Thus, market data vector x_t encapsulates various extracted market data information at time t together with a corresponding event that causes an update of the LOB status. The shape of the LOB is continuously evolving and the vector x_{t+1} at time $t+1$ is determined by the type of the event that is compressed in the vector x_t. The vector x_t is the result of simply merging the row containing data from the 'orderbook' file captured at the timestamp t with the corresponding row containing data from the 'message' file captured at the timestamp t. For each $0 \leq t \leq N$, where N is *the number of events during a trading day*, a market data depicted at a timeframe t is represented as a vector x_t. Thus, for any $0 \leq t \leq N$, the vector x_t has the form:
$x_t = (bidLevel_1, bidVolume_1, askLevel_1, askVolume_1, bidLevel_2, bidVolume_2, askLevel_2, askVolume_2, \ldots, bidLevel_n, bidVolume_n, askLevel_n, askVolume_n, Time, EventType, OrderId, Size, Price, Direction)$.

At this stage of the data reconstruction, we define the following data frame:

$$\mathscr{D} = \{x_t | 0 \leq t \leq \textit{ the number of events during a trading day}\}.$$

The idea is to classify each market data vector x_t into one of the labels from the set $S_+ = \{\text{buy}, \text{idle}\}$, if we consider semi-strategy emitting buy signals or into one of the labels from the set $S_- = \{\text{sell}, \text{idle}\}$, if we consider semi-strategy emitting sell signals. Note that the complete strategy can be obtained by combining both semi strategies. There are several ways to evaluate strategy, and a comprehensive overview can be seen in [93]. For the simplicity of the algorithm's learning procedure, the focus of this research is only on the semi strategy emitting the buy signals. Thus, the label is assigned as a Boolean function: 1 (true) indicates a buy order shall be issued, 0 (false) indicates a trader should idle. In preparation for employing machine learning techniques, such as the Gated recurrent unit networks (initially introduced in [28]) or Long short-term memory (LSTM) network (initially introduced in [64]) it is crucial to abstract relevant features from the LOB data. In order to gain more insights about the market behavior, apart from having features extracted from the LOB shape (such as open price, volume at the best ask price, etc.), we enhance the set by including also the standardly known technical analysis indicators.

3.6 The technical analysis indicators

Technical analysis market indicators are widely used for analyzing and forecasting the daily changes of stock indices. Trading stock indices is an important and demanding task for market participants. Technical indicators are based on mathematical transformations of price and each indicator delivers unique information. A comprehensive overview of the technical analysis indicators can be found in [87], [4], [58], [25]. Forecasting the direction of the daily price fluctuations with technical analysis can yield high financial benefits. It is important to emphasize that the technical indicators are derived by considering historical price data. Although the financial markets hold memory properties (see [127], [32], [92]), there is no guarantee that the technical indicators lead to an adequate prediction of a future price direction. Since the explanation of all technical indicators would exceed the scope of this thesis, just two of them are briefly presented below, while more on this topic can be found in [87],[4], [58], [25] and [30]. The technical analysis is the foremost approach in forecasting analysis for the short term, according to [86].

The Simple Moving Average: The Simple Moving Average (SMA) is a well-known indicator defined as the moving (running) average price calculated by adding the n recent prices and then dividing this total by n, where n is the number of the observed time periods. SMA belongs to the group of Moving Averages (MA) indicators, used to identify the potential price trend change and to describe the current trend. Each of the prices has equal weight, i.e.:

$$SMA = \frac{P_1 + P_2 + \cdots + P_n}{n},$$

where n is the number of observed periods and P_k is the price of an asset at period k, $k \in \{1,2...,n\}$. Short-term averages correspond to indicate quick changes in the asset price, on the other hand, long-term averages indicate slower changes.

The simple strategy based on MA is characterized as follows: if the stock price goes below the moving average line it indicates that it is a good time to sell, and vice versa. Furthermore, if two moving averages are considered, one with a long-term average and one with a short-term average, the moving average crossover strategy can be observed as follows. When the short-term average crosses above the long-term average, a so-called bullish moving average crossover occurs, indicating that it is a good time to buy. When

the short-term average crosses below the long-term average, a so-called bearish moving average crossover occurs, indicating that it is a good time to sell.

From the class of Moving Average(MA) indicators widely used, one is the exponential moving average (EMA). The EMA assigns larger weights on recent prices than on older prices, while the SMA assigns equal weight to each price.

Money Flow Index: Money Flow Index (MFI) is used to identify overbought or oversold conditions in an asset, i.e. buying and selling pressure. It is employed to show the inflow and outflow of money into security over time.

With the high price, the low price, the close price, and the volume of the observed interval, the following terms *typical price* and *money flow* (the total value of shares traded) are defined as:

$$typical\ price = \frac{high\ price + low\ price + close\ price}{3}, \tag{3.1}$$

$$money\ flow = typical\ price * volume. \tag{3.2}$$

When the price is higher than in the previous period, Money Flow is positive and it is added to Positive Money Flow. When the price is lower than in the previous period, Money Flow is negative and it is added to Negative Money Flow. Further, the *money ratio* is defined as:

$$money\ ratio = \frac{positive\ money\ flow}{negative\ money\ flow}. \tag{3.3}$$

Finally, the Money Flow Index (MFI) is defined as:

$$MFI = 100 - \frac{100}{1 + money\ ratio}. \tag{3.4}$$

The MFI generates value ranges from 0 to 100. A value of 80 is generally understood as a threshold and it is considered as buying pressure if an MFI value is above that threshold value, while a value of 20 is understood as a threshold above which is selling pressure. Note that buying pressure is when there are more buy orders than sell orders outstanding and thus, stock prices rise. On the other hand, stock prices fall in the case of selling pressure.

3.7 The data transformation

This section contains the data transformations used to prepare data for the research. More precisely, in order to extract the relevant information from the vast amount of data we dynamically reconstruct the LOB data and extract the features of interest. This includes the close price, technical analysis indicators, the number of executed bid orders, the best bid price, etc. The authors in [69] introduced a data reconstruction system and order flow visualization particularly for the LOBSTER data set. However, in this thesis, in order to proceed with the main part of the research, a customized technical pre-processing of the data is introduced and it represents an enhanced version of the approaches presented in [99] and [100]. Since for each trading ticker the limit order book is updating over one million events per day, there is an enormous amount of data. Hence, a technical pre-processing of data is employed in order to seize the applicable information from it.

The data transformation: The structure of the data transformation process consists of the following four stages:

1. *Raw aggregation*: In order to be able to extract the insights of relevant information, data aggregation over different time intervals is involved. Let $\mathscr{I}$ be a set of all considered time intervals, i.e. 60 second interval, 180 second interval, etc. For each $interval \in \mathscr{I}$ we define a complete partitioning over a single trading day data as:

$$\mathscr{P}_{interval} = \{\mathscr{T}_{j\cdot interval}^{(j+1)\cdot interval} | 0 \leq j < \lfloor \frac{trading\ day\ duration(in\ sec)}{interval(in\ sec)} \rfloor \}.$$

   Note that $\mathscr{T}_s^e$ is the part of a data identified by the start time s and the end time e, i.e. $\mathscr{T}_s^e = \{x | x \in \mathscr{D} \ \wedge \ x_{time} \geq s \ \wedge \ x_{time} \leq e\}$. For each selected time interval over which the aggregation is employed, various features (such as a number of canceled limit orders, a number of submitted limit orders, open price, close price, maximum price, minimum price, etc) are extracted to describe the market behavior during that time interval.

   Therefore, we introduce a set of functions used for the raw aggregation $\mathscr{A} = \{a_i\}_{i=1}^m$. This set $\mathscr{A} = \{a_i\}_{i=1}^m$ consists of m different functions such as extract the number of canceled orders, extract the number of submitted orders, extract open/close price, extract maximum/minimum price.

We iterate over the set $\mathscr{A} = \{a_i\}_{i=1}^{m}$, and each aggregation function a_i incomes the part of data $\mathscr{T}_s^e$ and outputs the $a_i(\mathscr{T}_s^e)$. Thus, the vector $y_s^e = (a_1(\mathscr{T}_s^e), \ldots, a_m(\mathscr{T}_s^e))$ is obtained by applying custom aggregation functions over the provided part of data $\mathscr{T}_s^e$. At this point parallelization within the Apache Spark can be applied due to the natural partitioning of the data set. More precisely, the computation can be simultaneously performed on separate parts of the data set, which makes the computational process significantly faster by the employment of the Apache Spark infrastructure.

The aggregation is done with respect to the intervals from $\mathscr{I}$. For each $interval \in \mathscr{I}$ and for each $0 \leq k \leq \lfloor \frac{trading\ day\ duration}{interval} \rfloor$, we define $y_k = y_k^{(k+1)\cdot interval}$. Now we can define the following chronologically ordered data set:

$$\mathscr{D}_{interval} = \{y_k | k = 0, \ldots, |\mathscr{P}_{interval}| - 1\}.$$

2. *Technical indicators*: Since technical indicators are widely used in the trading industry, we enhance our data set by incorporating the technical indicators in order to obtain relevant observations regarding market behavior.

   In order to compute the technical indicators an open-source library called *TA-Lib*[4] is used. This library contains the algorithms' implementations for the standardly known technical indicators needed to perform technical analysis of the financial market data.

   To proceed with including the indicators into our data set, a new partitioning $\mathscr{W}_{interval}$ is defined as:

$$\mathscr{W}_{interval} = \{(y_1), (y_1, y_2), \ldots, (y_1, y_2, \ldots, y_{|\mathscr{P}_{interval}|-1})\}.$$

   For each selected row in the data $\mathscr{D}_{interval}$ a new item in $\mathscr{W}_{interval}$ is generated and it contains chronologically all previous rows from $\mathscr{D}_{interval}$ including the currently selected row. By having this partitioning $\mathscr{W}_{interval}$ introduced we are in a position to compute the technical indicators using the set of inherent functions present in the *TA-Lib*. By $\mathscr{M} = (m_j)_j$ we define all functions contained in the *TA-Lib*. Thus, for any $w_i = (y_1, y_2, \ldots, y_i) \in \mathscr{W}_{interval}$ the output $m_j(w_i)$ is a technical indicator. Again, multiple computations can be run in parallel on separate parts of the data set. The

[4] Ta-lib: Technical analysis library. https://www.ta-lib.org/. Accessed: 2020-05-06.

data set consisting of technical indicators is defined by:

$$\mathscr{L}_{interval} = \{(m_1(w_i), \ldots, m_n(w_i))_i | w_i \in \mathscr{W}_{interval}, \wedge n = |\mathscr{M}|\}. \tag{3.5}$$

Further, let $\mathscr{F}_{interval}$ be the data set consisting of the information of the aggregated LOB enhanced by the technical indicators:

$$\mathscr{F}_{interval} = [\mathscr{L}_{interval}, \mathscr{D}_{interval}].$$

3. *Data lapsing*: During this stage, all different data sets obtained with respect to the different time interval length are merged into a single data set. More precisely, with aim to utilize the knowledge each interval is joined together with the related larger scaled interval. Thus, for $i, j \in \mathscr{I}$ and $i < j$, each vector of features from the $\mathscr{F}_i$ is merged together with the most recent vector of features from the $\mathscr{F}_j$ which had closed prior to the start of interval i. Therefore, at the end of this stage, we define the set $\mathscr{L} = \{\mathscr{F}_1, \mathscr{F}_2, \ldots, \mathscr{F}_n\}$, where $n = |\mathscr{I}|$. Note that due to a large number of extracted features this step is omitted on some occasions. Precisely, for this research the focus is on the particular data set $\mathscr{F}_{interval}$ that is obtained by setting the time aggregation interval to 180 *seconds*, i.e. $\mathscr{F}_{interval} = \{\mathscr{L}_{180}, \mathscr{D}_{180}\}$.

4. *Data labelling*: We consider semi-strategy emitting buy signals. Therefore, a label from the set $S_+ = \{\text{buy}, \text{idle}\}$ is assigned to each vector of a training set. Thus, with respect to the risk-reward ratio (RRR), we label each vector of the data set using the Boolean function. If the desired risk-reward ratio (RRR) is met, a value of 1 is assigned until the end of each trading day and a buy order should be issued. If the price goes below our risk level (stop-loss level) then a value of 0 is assigned and the indication should be to idle. The particular algorithm is presented below (see subsection 3.7.1). The algorithm is run with the following values $REWARD = 0.08$ and $RISK = 0.04$, and for the particular data set $\mathscr{F}_{interval}$ that is obtained by setting the time aggregation interval to 180 *seconds*. Note that this algorithm in a slightly different form is presented in [100] and [99].

3.7.1 The sweep forward algorithm

Algorithm 1 The sweep forward algorithm that assigns label to each data vector.

Input: $\mathscr{F}_{180}$, $RISK$, $REWARD$
Output: the labelled data set $\mathscr{F}^*_{180}$

$\mathscr{F}^*_{180} = None$
for $index = 0$ to $|\mathscr{F}_{180}|$ **do**
  $initialPrice = getPrice(\mathscr{F}_{180}[index])$
  $datavector = \mathscr{F}_{180}[index]$
  $stopLoss = initialPrice \cdot (1 - RISK)$
  $targetReward = initialPrice \cdot (1 + REWARD)$
  $label = False$
  for $forwardIndex = index + 1$ to $|\mathscr{F}_{180}|$ **do**
    $currentPrice = getPrice(\mathscr{F}_{180}[forwardIndex])$
    $temporaryStopLoss = currentPrice \cdot (1 - RISK)$
    if $(temporaryStopLoss > stopLoss)$ **then**
      $stopLoss = tempStopLoss$
    end if
    if $(currentPrice < stopLoss)$ **then**
      $\mathscr{F}^*_{180}.append(labelPointAsIdle(datavector))$
      $label = True$
      break
    end if
    if $(currentPrice > targetReward)$ **then**
      $\mathscr{F}^*_{180}.append(labelPointAsBuy(datavector))$
      $label = True$
      break
    end if
  end for
  if $(label == False)$ **then**
    $\mathscr{F}^*_{180}.append(labelPointAsIdlePoint(datavector))$
  end if
end for
return $\mathscr{F}^*_{180}$

3.8 Fourier transformation based features

Even though *TA-Lib* contains a large set of standardly known technical indicators, new technical indicators utilizing Fourier transformation are incorporated in this research.

The order book is updating up to 18 quadrillion messages per day, requiring nanosecond precision. Because of the mentioned frequencies, market behavior has become seemingly harder to model.

Since the price and the number of orders sitting at each LOB level can be perceived as a discrete signal over time, the idea is to employ advanced signal processing techniques to extract new features. Every time series can be decomposed onto the series of sines and cosines (for more details see [74] or [19]). Therefore, the idea here is to decompose the signal from the data produced by the stock market in order to uncover some latent properties that were not visible before.

The Fourier transform has many applications in different fields, especially physics, engineering, and applied mathematics. For example, in [130] the author proposed a new kernel-smoothed fast Fourier transform technique for pricing American options under stochastic volatility and showed a very good performance of the technique. In [49] the authors established conditions that ensure the existence of the Fourier transform valuation formulas in a general framework.

In particular, proposed features are built on the assumptions that the *price* (e.g. high price, close price, best ask price, etc) can be decomposed into its true *value* plus some *additive* component. Further, that *additive* component is modeled with the Price Oscillator.

The Price Oscillator is very similar to the Moving Average Convergence Divergence (MACD). The Price Oscillator (PO) is defined as the difference between the moving averages: one shorter period and one longer period.

Therefore, we interpret the Price Oscillator as a difference between the observed close price signal (interpreted as the moving average of it with period $n = 1$) and the moving average of it with period $n = 5$:

$$PO = price - movingAverage(price, 5). \tag{3.6}$$

The particular concept of the Fourier transformation based features extraction process is presented in Section 4 in [100].

3.9 The proposed methodology

Let $\Phi = \{f_1, f_2, f_3, \ldots, f_n\}$ be the set of all extracted features. Note that the set Φ consists of the features extracted from the order book during the data transformation part including the technical indicators present in the library *TA-Lib*, and further enhanced with the Fourier transformation based features (introduced in Section 3.8). The set Φ contains a vast number of features, so if we plug all the features from Φ into the GRU network model, the model tends to overfit. The overfitting occurs when the model is built by using a limited set of data thoroughly and completely and thus it may not fit additional data, i.e. the model may fail to predict future values accurately (see [115] and Chapter 7 in [33]). Therefore, the starting mission in this research is to distinguish relevant features from those that hold less information when it comes to deciding if the position should be opened. The buy semi-strategy, a strategy emitting *buy* or *idle* signals, is considered. More precisely, the aim is to choose a subset of features $\Phi_1 \subset \Phi$ to improve the performance of the model of the proposed buy semi-strategy measured with respect to the F_1 score. Since each feature provides unique information, the idea is to navigate through this huge vector space of the extracted features towards those that could be beneficial in making a decision if the position should be opened or not.

Different approaches to perform feature selection have been explored in the literature. The authors in [77] compared four different models, based on different feature selection methods, which use fifty-five technical indicators as input variables to predict the direction of the price. Moreover, hybrid Artificial Neural Network (ANN) models were employed in [59] to choose technical indicators that are relevant to determine stock market price direction. However, note that the feature selection approach presented in this theses differs from the approaches presented in the aforementioned literature, and it is devoted to select features extracted by the customized data transformation (presented in Section 3.7).

The first part of this section is devoted to presenting the theory that stands behind the proposed methodology of this research.

3.9.1 Stochastic Universal Sampling

Stochastic Universal Sampling (SUS), initially introduced in [8], is a single-phase algorithm that performs a selection from the population in genetic algorithms. A genetic algorithm represents a heuristic search method that uses principles of natural evolution.

In genetic algorithms (see [65], [55]), genetic operators perform actions on the population of individuals and mimic the natural evolution process (inheritance, mutation, selection, and crossover). The main advantages of the SUS compared to other genetic algorithms' selection operators are reduced bias and increased efficiency.

The SUS algorithm performs selection using a single random value sample by choosing N individuals at evenly spaced intervals. Thus, individuals are selected with respect to their fitness, however, individuals with lower fitness values still have a chance to be chosen. Firstly, it is assumed that each individual has a fitness value and that the summarized fitness of all individuals is denoted by S. Note that the individuals are sorted descending with respect to their fitness values. Then, each individual is mapped to an adequate contiguous segment of a line within the interval $[0,S]$, such that the length of the segment is equal to the fitness values of that individual (as depicted in Figure 3.2). In order to select N individuals, a value from $(0,S/N)$ interval is randomly selected and the individual at that position is selected. Then, the next individuals are selected at each position obtained by incrementation of the current position by S/N (this is repeated $N-1$ times). This is depicted in Figure 3.2.

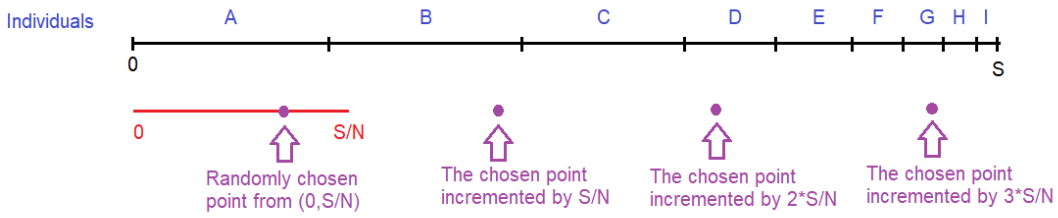

Figure 3.2: *The SUS algorithm aims to choose $N=4$ individuals from the population $\{A,B,\ldots,I\}$. Assume that the individuals are sorted descending with respect to their fitness values and each individual is placed on the $(0,S)$ line taking exactly the same length as its fitness value. One number is randomly chosen from $(0,S/N)$ and the first selected individual is A. After incrementation of that point by S/N three times, each time one new individual is selected (i.e. in particular B, D, G in this Figure).*

3.9.2 Gated Recurrent Unit

In order to present the Gated Recurrent Unit, a brief explanation of neural networks and specifically recurrent neural networks is given. Artificial neural networks (ANNs or simply NNs) represent a set of algorithms intending to recognize patterns. The modeling of these group of algorithms is inspired by the biological neural networks in animal and human brains. The NNs are constructed to recognize patterns and find structure within data that is usually complex and imprecise. The neurons in NNs are very simplified abstractions of biological neurons and they need to be trained to perform useful functions (see [42]). How these neurons are connected is crucial for the performance of the neural network. These neurons are organized into a set of layers and there are three classes of layers: input, hidden and output. The nodes from the input layer do not perform any computation. The task of these so-called input nodes is to provide information to the hidden nodes. The hidden layers of neurons are located between the input and output layers. The hidden nodes perform computation on the information acquired from the input layer and pass new information to the output layer. The output nodes transform the information obtained from the hidden layer and transfer the output. A simple neural network is depicted in Figure 3.3, where there are two hidden layers.

A deep neural network (DNN) is an artificial neural network with multiple hidden layers [11]. Studies showed that increasing the number of hidden layers increases the accuracy of neural networks (see [60]). The reader is referred to [106] for an overview of deep neural networks.

Adding additional hidden layers is beneficial when nonlinear mapping is used. On the other hand, adding additional hidden layers in the case of linear mapping is useless since the composition of two or more linear functions is itself a linear function. Those nonlinear functions are called activation functions and some of the examples are: a step function (e.g. a sign function), a sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$), a hyperbolic tangent function ($tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$), etc. Each neuron takes input x and produces output y by performing a calculation that is usually $y = f(wx+b)$, where w and b are both the neuron's scalar parameters that are being adjusted during the training phase, while f is an activation function. More on neural networks can be found in [10], [42], [5], etc.

The training process of a neural network can conceptually be described as in [72] and it is briefly presented here within the following steps:

1. The input is introduced to the neural network.

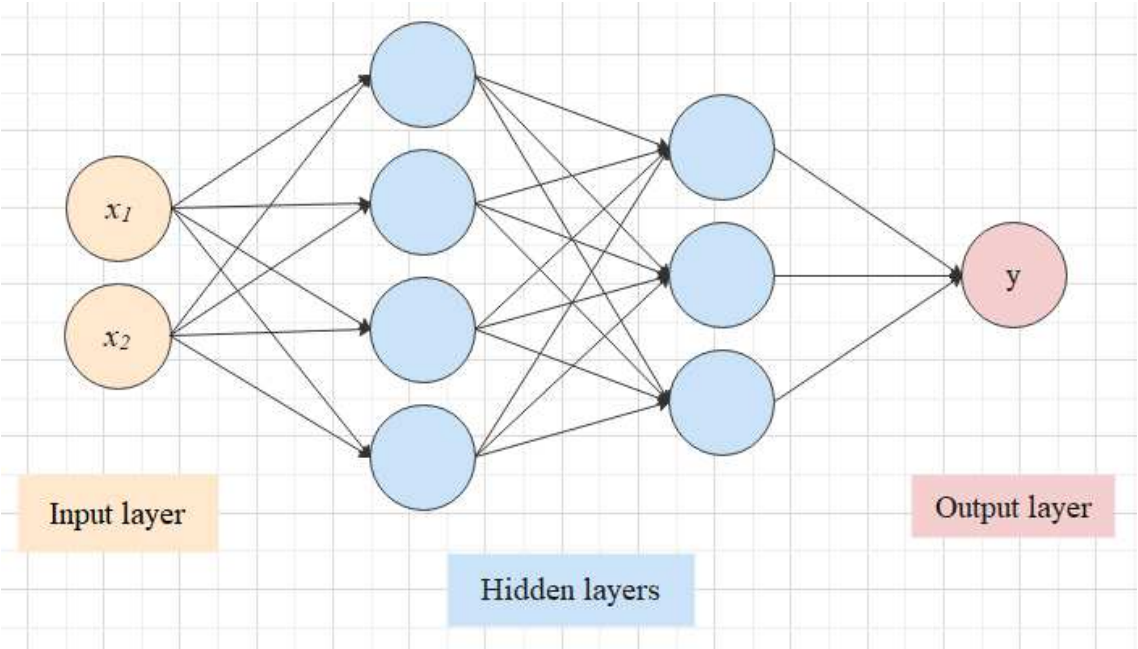

Figure 3.3: *A simple artificial neural network with two hidden layers*

2. All the weights of the network are assigned some default value.
3. An output is obtained by passing the input through neurons in each layer.
4. The generated output is compared with the expected output or label.
5. The error between the prediction and label is then used to update the weights of each node.
6. The error is then propagated in backwards direction through every layer, to update the weights in each layer such that they minimize the error.

The training phase based on the backpropagation can be performed by applying one of the following methods: Online (stochastic) method and Batch method (see [72]). In the case of online or Stochastic learning, a single sample is fed to the network and the weights are updated according to the output error. The optimization method most commonly used to update the weights is called stochastic gradient descent (SGD) method (see [17]). The samples are randomly taken from the whole set of data, and thus this method is called stochastic. Note that the converge to the desired accuracy level can in some cases happen before all the samples are used.

Batch learning is the method when the dataset is divided into several batches (parts). To compute the error and update the weights, all available batches are passed into the neural network. Gradient descent [104], as an iterative process, is used to optimize the learning rate. One batch is used for each iteration and the batch size is the number of training examples in a batch. Note that the higher the batch size, the more memory space is needed. One epoch is when an entire dataset is passed forward and backward through the neural network only once. Usually, multiple epochs are used to obtain the algorithm's convergence. The number of batches is equal to the number of iterations for one epoch. Note that passing the entire training dataset into a neural network at once is not possible in some cases.

As stated in [60], large training sets are necessary for good generalization, but large training sets are also more computationally expensive, which represents a common challenge in machine learning applications.

In the neural network depicted in Figure 3.3, the information moves only forward, from the input nodes, through the nodes in the hidden layer (or more hidden layers) and to the output nodes. This kind of neural network that contains no cycles, called a Feedforward neural network, was the first and simplest artificial neural network proposed. In the case of a feedforward network, there is a single input layer and a single output layer, while there can be zero, one or multiple hidden layers. Feedforward neural networks have been mostly used for pattern recognition problems.

In the contrast to the Feedforward neural networks, Recurrent neural networks (RNNs) are a class of neural networks that remember the past (i.e. previous outputs) and the past has an influence on future decisions (previous outputs can be used as inputs). Thus, RNNs are more conducive when dealing with sequential data types (e.g. time series, natural language). A possible representation of a simple recurrent neural network can be seen in Figure 3.4. For a given sequence $x = (x_1, x_2, ..., x_N)$, the RNN updates its hidden state $h_t = \phi(h_{t-1}, x_t)$ by applying a nonlinear function ϕ (see [29]), while h_0 is initially set to 0. In addition to the final output, the RNN may also produce an output $y = (y_1, y_2, ..., y_N) = (h_1, h_2, ..., h_N)$ i.e. the hidden state output for each input time step is returned (as presented in Figure 3.5). Note that Figure 3.5 contains the representation that is unrolled for the sake of understandability and it is used for further explanations of the recurrent data flow. Recurrent neural networks (RNNs) have already been used as a powerful tool in modeling LOB dynamics (e.g. see [46] and [45] as mentioned in Section 3.2).

In the case of the basic RNN, also called Vanilla RNN, Bengio at al. [12] indicated the

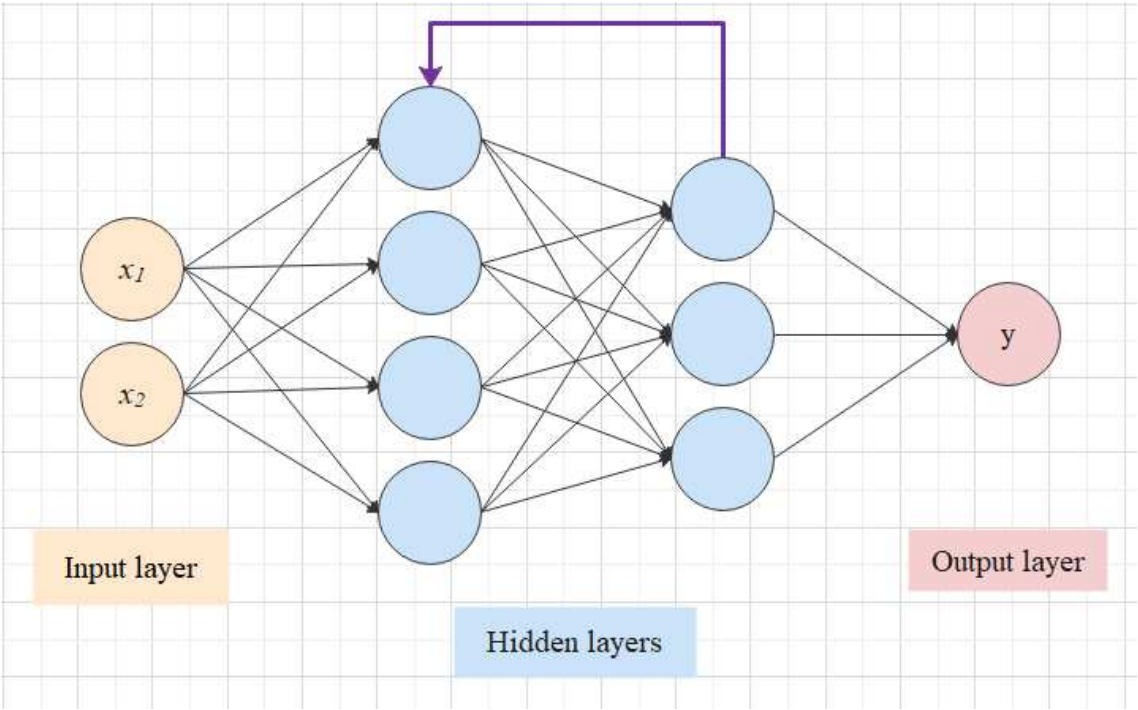

Figure 3.4: *A simple recurrent neural network*

vanishing gradient problem. Long-term dependencies are difficult to record if the Vanilla RNN is used since the (stochastic) gradients tend to either vanish or explode when working with long sequences (for more details see [12]). To solve this problem, two models were created with their internal mechanisms called gates that can regulate the flow of information. Thus, they use more computational resources compared to basic RNN.

One of the widely used RNNs in the field of deep learning (see [106], [60] and [5]) is Long short-term memory (LSTM) neural network. LSTM was initially introduced in [64], as a novel, efficient, gradient based method created to improve the performance of the previous RNNs. A LSTM cell contains operations that allow the LSTM to choose which information to remember or forget, due to its gates, as depicted in Figure 3.6, which contains a representation of the LSTM model as a modification of the RNN depicted in Figure 3.5 since the control flow is the same, but the LSTM cell's operations are different.

Gated Recurrent Unit (GRU), introduced in [28], is a variant of the RNN architecture. While both the LSTM and GRU networks have the ability to seize long-term and short term dependencies, some of the main advantages of the GRU networks are that they perform faster and employ fewer parameters. These are among the main reasons for choosing

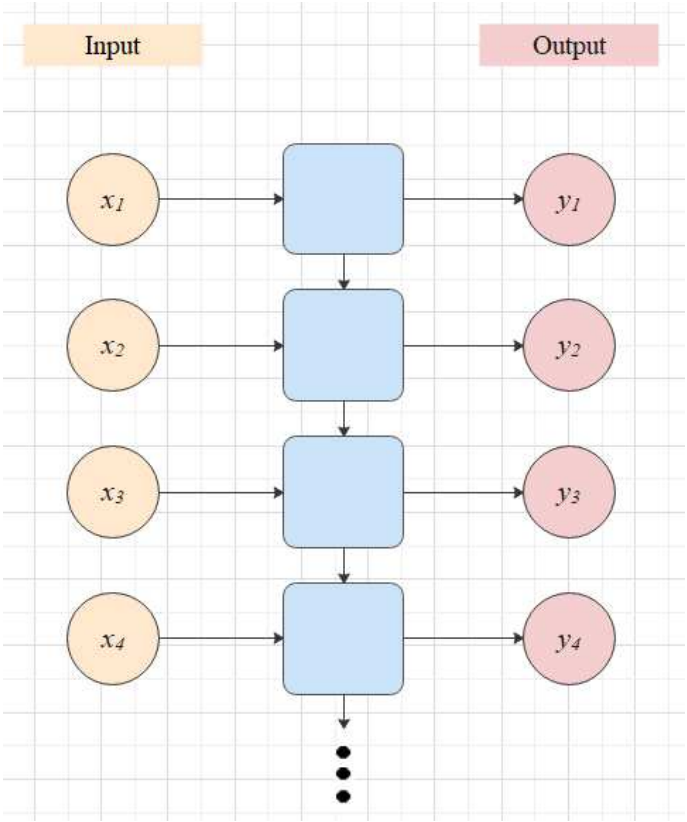

Figure 3.5: *A recurrent neural network's control flow is presented, i.e. the RNN processes the data sequentially passing on the information as it propagates forward (downwards as presented here).*

the GRU for this research. Note that, however, the LSTM neural networks overperform the GRU in some cases (for example, see [121]). The authors in [29] compared different types of recurrent units in recurrent neural networks (RNNs), especially the LSTM and GRU. The experiments on the tasks of polyphonic music modeling and speech signal modeling showed that GRU is comparable to LSTM, and both performed better than tested more traditional recurrent units. Since both the LSTM and GRU overperformed the other in certain tasks, as shown in [29], it can not be concluded that one of these two models is better in general.

A GRU cell can be seen in Figure 3.7, where the product is the Hadamard product (element-wise multiplication of two matrices of the same dimensions), σ is a sigmoid function and tanh is a hyperbolic tangent function. In the case of GRU architecture, gating mechanisms are used to control and manage the flow of information between cells, i.e. in

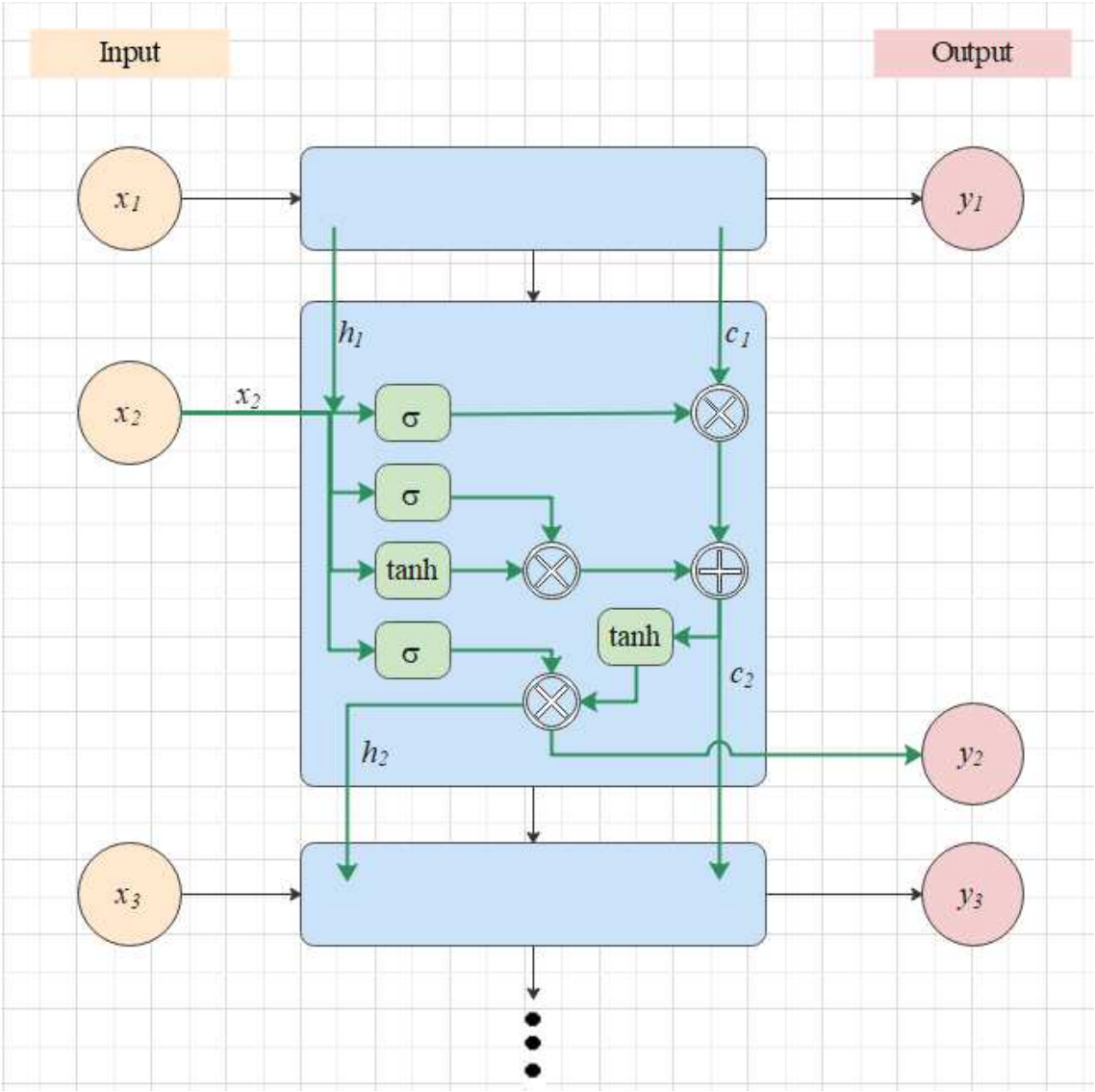

Figure 3.6: *A simple representation of Long short-term memory architecture*

order to solve the vanishing gradient problem of the Vanilla RNN, the GRU cell contains two gates: update gate and reset gate. Firstly, in order to calculate the update gate, the following formula is used:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}),$$

where σ is a sigmoid function, x_t is the input vector at the cell t, h_{t-1} is the output vector from the previous cell, while W and U are weight matrices to be learned. The update gate

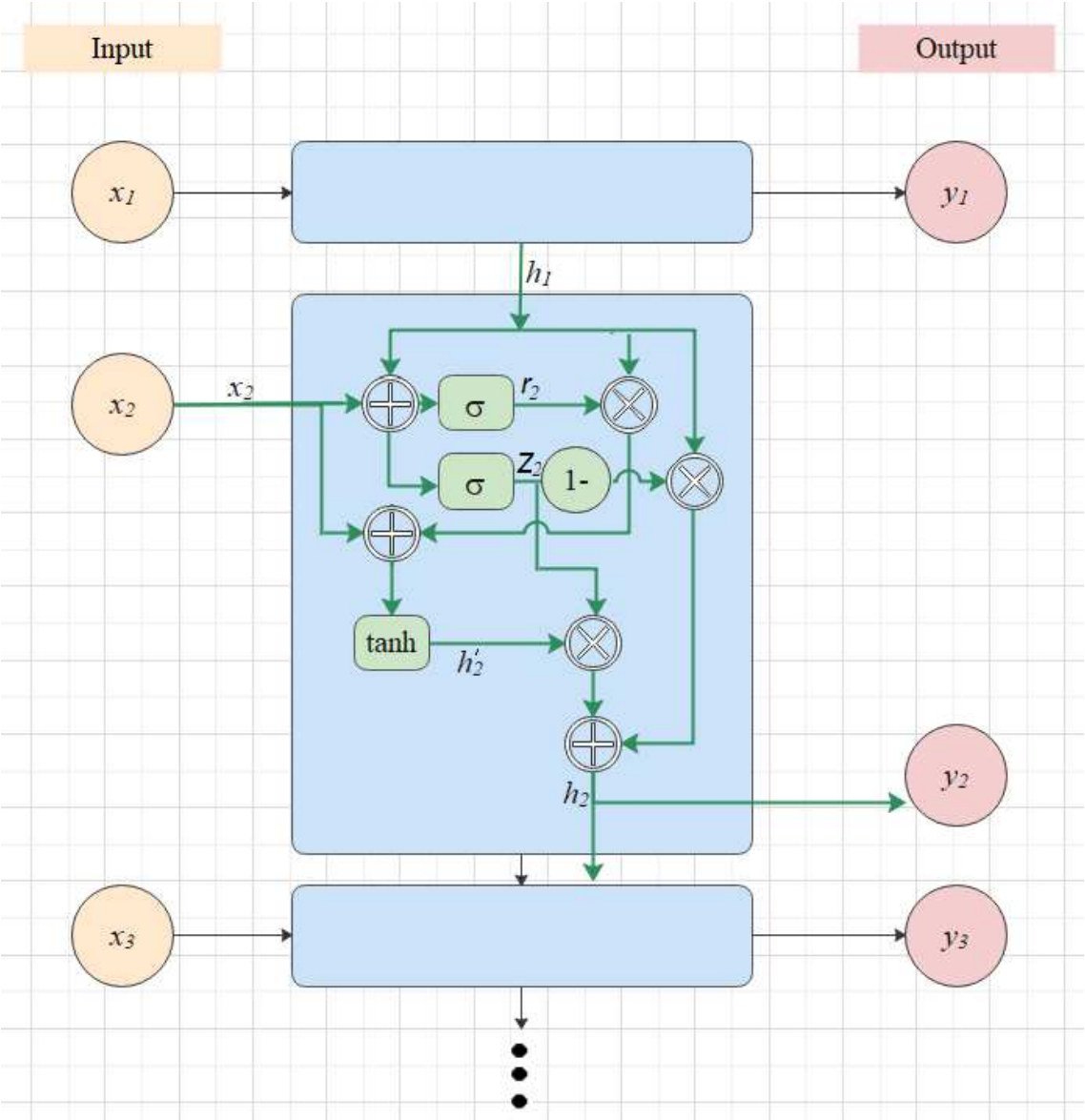

Figure 3.7: *A representation of Gated Recurrent Unit architecture*

z_t determines how much of the information from the previous steps to forward further. This date is crucial for preventing the vanishing gradient problem. Moreover, the reset gate r_t decides how much of the past information to forget and it is defined as follows:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}).$$

Further, a new memory content to store the relevant information from the past is calculated

as:

$$h'_t = tanh(Wx_t + U(r_t \odot h_{t-1})),$$

where $\odot$ is Hadamard product. Finally, the value h_t is calculated to be passed further:

$$h_t = z_t \odot h'_t + (1 - z_t) \odot h_{t-1}.$$

For more information on the GRU neural networks, the reader is referred to [28], [43] and [105].

There have been a variety of successful applications of the GRU to different problems. For example, the authors in [114] applied the review mechanism on the GRU proposing a new n-Gated Recurrent Unit with Review (nGRUR) model to perform answer selection. Experimental results in [48] showed that the GRU model overperformed other tested models in order to predict daily Bitcoin prices. Moreover, there are applications in robotics. For example, the authors in [23] presented a successful GRU based deep recurrent neural network to generate natural language descriptions for the scene a robot observes. There are also applications in the field of medicine, such as for machine health monitoring systems (see [126]). Other applications of the GRU architecture include wind power forecasting [44], sentence semantic classification [124], etc.

3.9.3 The GRU model architecture

In order to depict the time dependency among different points in time of the market signal, a model based on the GRU topology was utilized (see Table 3.4). The implementation is done by using the *Tensorflow*[5] library, while the basic technical details on the proposed GRU model are given here and summarized in Table 3.4.

The proposed GRU based model consists of three parts:

1. The GRU layer of 50 units is the layer that processes extracted features from every market vector. Note that each market vector has a dimensionality of 200, i.e. each market vector contains the values of 200 features. The GRU is set to see the previous 64 market vectors. In order for the GRU to also return the entire sequence of outputs for each sample, the parameter *return_sequences* is set to true.

[5] https://www.tensorflow.org/

Table 3.4: *Summary of the proposed GRU based model with respective input and output dimensions.*

Layer name	Input shape	Output shape	Explanation
GRU layer	(batch_size, lookback_period, 200)	(batch_size, 64, 50)	The input contains of 200 features with the lookback period of 64 market ticks
Flatten layer	(batch_size, 64, 50)	(batch_size, 3200)	Flattens the output of the prior step to be used as input for the dense layer
Dense layer with Sigmoid activation	(batch_size, 3200)	(batch_size, 1)	Outputs the final probability, whether the trade is suitable to open up the position

2. The Flatten layer is the layer that flattens the outcome of the GRU layer such that the data can be fed to the dense layer. The Flatten layer in *Keras*[6] reshapes the tensor to have a shape that is equal to the number of elements contained in the tensor. Note that this has no effect on the batch size.

3. The Dense layer is the final densely-connected NN layer receiving a vector of size 3200 and outputting a single value between 0 and 1. The Dense layer is the most frequently used layer from *Keras*. The output shape of the Dense layer depends on the number of units specified in the Dense layer. This layer uses the *Sigmoid* activation function and the returned value represents the probability that the strategy shall issue a *buy* command (if the returned probability is ≥ 0.5) or *idle* otherwise (the case when the returned probability is < 0.5).

3.9.4 Basic measures used in the proposed features selection methods

Mutual Information: The mutual information (MI), also known as the information gain, of two random variables, is a quantity that measures the amount of information about one random variable which is communicated with another one.

Formally, the mutual information of a pair of two random variables X and Y with values in $\mathscr{X}$ and $\mathscr{Y}$ respectively, with the joint distribution $p_{X,Y}$ and the marginal distri-

[6]Keras is an open-source neural-network library written in Python, build on top of TensorFlow 2.0. See: https://keras.io/ (accessed in May 2020).

butions p_X and p_Y, is calculated by the formula:

$$I(X;Y) = \sum_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_X(x) p_Y(y)}.$$

For a deeper theoretical overview of the mutual information, the reader is referred to [116]. Furthermore, a comprehensive overview of feature selection methods based on mutual information is stated in [119].

Cross-correlation: Cross-correlation measures the similarity between two signals, as a function that tracks the movements of one relative to the other. Furthermore, in time series analysis it is standard practice to normalize the cross-correlation function to take values in the interval $[-1,+1]$, i.e to use the Pearson correlation coefficient, the Spearman correlation coefficient, etc. The Spearman correlation coefficient measures the degree of a monotonic relationship between two random variables, while the Pearson correlation coefficient measures the degree of linear correlation between two random variables.

Autocorrelation: Autocorrelation measures a correlation of a signal with its lagged version over different successive time periods. Informally, it is a degree of similarity between a signal's present value and its former values.

3.9.5 The proposed features selection methods

To perform a feature selection, the first question is how to determine the quality of any considered feature.

Here, the following two hypotheses are assumed:

1. A higher absolute value of the feature autocorrelation means a higher likelihood of the indicator imposing a repetitive pattern thus we can infer knowledge easier.

2. Another hypothesis is that the feature having a higher value of mutual information with respect to the close price holds more information pertaining to the decision if the position should be opened or not.

For the feature selection process, to each feature a *SCORE* value is assigned with respect to the the following two measures:

1. The feature's absolute autocorrelation value.

2. The mutual information between the feature and the categorical variable of the close price.

In order to determine the optimal lag, i.e. time-period with respect to which we calculate autocorrelation of each feature, we observe the trade duration distribution. Most of the trades' durations fall between 22 and 40 time-periods. Note that one time-period is three minutes long, as we employ data aggregation over 180 seconds time interval, i.e. 40 time-periods is equal to 120 minutes. Thus, in order to calculate a score value for each feature, the following calculation is performed. For each lag from the $[22, 40]$ interval the autocorrelation value is calculated and then the average value from that set of the autocorrelation values is assigned as a score value for the corresponding feature.

Regarding the second hypothesis, given a feature X (e.g. moving average), we want to compute mutual information between feature X and the close price. Each close price value is a discrete value from the broad range and therefore it potentially can happen that $P(close\_price = C)$ converges to 0, given that C could be the unique price of the stock that never repeats itself.

In order to mitigate this situation when many stock close prices would have a marginal probability of 0, rather than computing mutual information between X and the close price itself, we fit 400 clusters via the K-means algorithm (see [75]) into the close price series. Therefore, we assign each close price value efficiently into one of 400 clusters. We treat this as a new categorical variable, and rather than computing mutual information between the close price and the feature X, we compute the mutual information between feature X and our newly introduced categorical variable.

Once a SCORE value is assigned to each feature, Stochastic universal sampling (SUS) is implemented in order to select features with respect to their score (higher the score, higher the likelihood for the feature to be selected).

3.9.6 The groups of features

Firstly, the following two groups of features are considered:

1. The old-fashioned features - consisting of the features extracted from the LOB and standardly known technical indicators (see Section 3.7).

2. The set of features enhanced with the Fourier transformation based features is the set of features that contains old-fashioned ones and features that employ Fourier transformation (introduced in Section 3.8).

The whole set of features enhanced with the Fourier transformation based features consists of 5996 features, thus the SUS algorithm is proposed to select 200 features proportionally to a feature *SCORE* value.

We repeat measurements on top of each feature set (employing the randomized SUS selection) ten times and then apply statistics to evaluate the results.

The idea is to evaluate if the performance of the GRU network model is improved when features are selected with respect to the newly proposed methods.

Please note that none of the observed models leads to high classification accuracy. That is completely expected due to the specific behavior of the stock market, i.e. a lot of noise present in the data. However, we aim to compare the performance of the model when fed with different groups of features.

3.9.7 The F_1 score

The confusion matrix (see [88]) is one of the most used metrics to measure the performance of a classification based supervised learning models. Two widely used metrics are precision and recall (e.g. see Section 4 in [129]), defined as presented with equations (3.7) and (3.8):

$$precision = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{3.7}$$

$$recall = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \tag{3.8}$$

These two metrics are different in a way that precision calculates how accurate the model is in the sense of how many of the predicted positive are actually positive, while recall measures how many of the actual positives are predicted to be positive. Thus, precision is an adequate measure to use when the cost of a false positive is high, while recall shall be used when the cost of a false negative is high. In this research, a measure called F_β score is used and it is a function of precision and recall (see Eq. (3.9)). Note that the more we decrease β, the more we prefer precision over recall. Conversely, the more we increase β, the more we favor recall over precision. For example, the F_β-measure with a smaller β value, such as the $F_{0.5}$-measure, is utilized when more weight is put on precision, and

less weight is put on recall, i.e. when minimizing false positives is more important than minimizing false negatives. On the other hand, F_β-measure with β value higher than 1 (e.g. the $F_{1.5}$-measure) is utilized when minimizing false negatives is more relevant than minimizing false positives, i.e. when more attention is put on raising the importance of false positives rather than false negatives. The F_1 score is a very convenient measure if we want a balance between precision and recall (see [88]).

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{3.9}$$

3.10 Measurements and performance

This section is devoted to an evaluation of the informativeness of newly suggested features and the capability of the proposed methods for the features selection. In order to seek a balance between precision and recall, to measure performance we compute the F_1 score (the harmonic mean of precision and recall). Since the SUS based method is used we repeat measurements ten times and compute the mean and standard deviation of the F_1 metric.

The proposed feature selection approach is applied and the model is trained using the *Tensorflow* defined GRU model - on top of the train and validate parts of the datasets. The test split of the respective dataset is never presented to the model during training and this is the part of data used for the evaluation with respect to the F_1 metric score. This guarantees unbiased results and proves that the model generalizes well on previously unseen data.

The aim is to confirm the following hypothesis:

"Extracting the Fourier based properties of the stock market signal and employing the SUS with respect to the feature's assigned SCORE value to choose the features, significantly improves the F_1 score of the GRU model".

Firstly, for the benchmark model, we use the dataset consisting only of the old-fashioned features. Then, we use the dataset with both old-fashioned features and the Fourier transformation based features. We split either of the datasets into three parts: Train (60% of data), Validate (20% of data) and Test (20% of data).

3.10.1 Fourier transform based features hypothesis

The performance of the GRU model (see Table 3.4) when fed with the Fourier transformation based features and the performance of the same model fed with the old-fashioned feature set only are compared.

Having the whole training set labeled as described in the paragraph 'Data labelling' in Section 3.7, a GRU model is trained in order to predict the respective label. The aim is to show that employing Fourier transform based features significantly improves the predictive capabilities of the proposed GRU model. In order to do so, we run a standard *p-value* significance test, where we assume the following null hypothesis H_0:

"Fourier transform based features do not improve the performance of the proposed GRU model when measured by the F_1 score".

Proposed procedure

The following procedure is proposed:

1. The first step is establishing a benchmark GRU model that assigns a label to the market vector by employing the only the old-fashioned features (no Fourier transform based ones). In order to maximize the accuracy, only a subset of features is selected by assigning the *SCORE* value and then the SUS algorithm is run to select the most appropriate features (the higher SCORE value more likely for the feature to be selected). Since the SUS algorithm is utilized, the process is repeated multiple times to obtain a good statistical sample. As a key performance indicator, the F_1 score is measured, more precisely its mean and standard deviation.

2. New Fourier transformation based features are proposed and implemented as part of every market vector data.

3. Then, the measurement is re-run as described in the first step, but with the difference that for this step the data set additionally contains Fourier based transformation features.

4. Finally, the mean values of the F_1 score obtained for the measurement of the model containing the proposed Fourier transformation features (from step 3) is compared with the benchmark model trained on top of the data-set not containing the Fourier transformation features (from step 1).

Feature set	Ranking method	F1 mean score	F1 std
With new features	Mutual information	0.126441091	0.085182755
	Autocorrelation	0.070379998	0.064926468
Old features only	Mutual information	0.023346747	0.009447132
	Autocorrelation	0.01543692	0.005528859

Table 3.5: *F1 mean score and F1 standard deviation after running ten times the measurement in respective feature set and ranking method category (40 measurements overall).*

Ranking method	z-score	p-value	Significant ($\alpha = 0.01$)?
Mutual information	3.827217641	6.479996E-05	Yes
Autocorrelation	2.676031416	0.003724983	Yes

Table 3.6: *Table containing z-scores and p-values for respective category with the significance information.*

A summary of our measurements along with the observed mean and standard deviations of *F1* score can be found in Table 3.5. The first column of Table 3.5 distinguishes the feature set used (the Fourier transformation based features and the old-fashioned features), while in the second column the ranking method is given, i.e. the method used for generating the *SCORE* value. The third and fourth columns contain *F1* mean and *F1* standard deviation value in each of these groups of experiments. Furthermore, we compute respective *z-scores* for our measurements along with respective *p-values*, see the second and third column of Table 3.6. As can be noted in the fourth column of Table 3.6, our new features yield to significantly improved performance in our benchmark GRU model (for both *SCORE*-ing methods), with respect to the significance level of 0.01.

Therefore, we can safely reject our null hypothesis and affirm that proposed Fourier transform based features improved the model performance significantly.

3.10.2 Cross-comparing the proposed feature selection methods with the random choice of features

The two feature quality metrics that determine *SCORE* values are cross-compared, namely:

1. The mutual information between the feature and the categorical variable of the close price.

2. The feature's absolute autocorrelation value.

Ranking method	**z-score**	**p-value**	**Significant** ($\alpha = 0.05$)**?**
Mutual information	2.165600285	0.015	Yes
Autocorrelation	0.110756942	0.456	No

Table 3.7: *Z-score and p-values of singificance test comparing random selection with SUS and mutual information or autocorrelation scoring.*

We want to investigate if the simple random feature selection is outperformed by the stochastic universal sampling with either mutual information or auto-correlation feature ranking methods. We assume the following null hypothesis $\widetilde{H_0}$:

"Random selection performs as good as either of the proposed methods (mutual information or autocorrelation ranking accompanied with the SUS algorithm)".

We measured the mean value of the F_1 score when using the random feature selection instead of the SUS algorithm with a scoring method, and observed *z-score* and *p-values* can be found in Table 3.7. As can be noted from Table 3.7, we can reject the hypothesis $\widetilde{H_0}$ since random selection underperforms the SUS algorithm when used with the mutual information at the significance level of 0.05.

However, we observe that when the score value is assigned with respect to the feature's auto-correlation, the Stochastic Universal Sampling is performing better in the average, but it is not statistically significant.

Note that, the presented experiment was performed with the Apple stock data. However, the model shall scale-out well also to other stocks as their prices can be perceived as discrete signals. The biggest improvement of the GRU model performance is expected to be shown in markets whose participants are more automatized.

4

Conclusion

This chapter is devoted to summarizing the results of this thesis and indicating possible extensions.

In Chapter 2 a simple, but non-trivial, limit order book model is considered. The new orders get placed at distance $\mu \geq 1$ above the mid-price and a trade event occurs whenever the mid-price reaches the price level on which there is some volume. The special interest is on the avalanche approach, motivated by the fact that the orders in the limit order book accumulate on some levels and get executed when the mid-price process crosses such values. Further, two execution mechanisms for the ask side of the book are considered, namely Type I trade and Type II trade execution mechanisms. If the limit order book is initially full, a Type I trade occurs whenever the mid-price increases compared to the price on which the previous trade had occurred. On the other hand, when the order book is initially empty, the first Type I trade occurs at the price level k, where $k \in \{1,2,...,\mu\}$, if the mid-price goes down by $\mu - k$ steps, but not below, and then goes up to the level k (see Figure 2.13). After the first trade occurs (either a Type I or a Type II trade) in an initially empty order book, any subsequent Type I trade happens when the mid-price increases (as in the case when the order book is initially full). If the order book is initially full, but also if it is initially empty, a Type II trade occurs when the mid-price falls down by μ or more and then goes up by μ. Specifically, flash crashes are special kind of Type II trades that occurs in a very short time interval (less than $\varepsilon > 0$). In section 2.5 the results are derived with an assumption that the order book is initially empty, and it is emphasized that after the first trade (either a Type I or a Type II trade) occurs, the analysis of the avalanche length is equivalent as in the model when order book is initially full.

With the equation (2.8), the best ask process is described by the mid-price process without reference to the full order volume process. The best ask price process stays inside the zone bordered by the mid-price and the mid-price plus $\mu+1$, i.e. $S_n \leq \alpha_n \leq S_n+\mu+1$ for each $n \geq 0$. Further, if the price level u is higher than the current best ask price process at the time $n \geq 0$, than there is some volume at the price level U at the moment n. Even more, there is some volume at price level u at time n, $n \geq 0$, if and only if $u \geq \alpha_n$, see Lemma 2.3. The trade event occurs exactly when the mid-price reaches the best ask price, see Figure 2.5. Therefore, as stated in Corollary 2.2, the trading times can be defined as

$$\tau_0 = 0, \quad \tau_i = \inf\{n > \tau_{i-1} : S_n = \alpha_n\}, \quad i \geq 1. \tag{4.1}$$

Lemma 2.1, establishes the result for a strict ascending ladder time of the random walk, and thus has motivated us to introduce the simplified trading times $\{\rho_i : i \geq 0\}$ and the simplified intratrading times $(R_i)_{i\geq 1}$ (see Definition 2.3). The probability generating function for the simplified avalanche length $E[z^{L_\varepsilon}]$ is given by

$$E[z^{L_\varepsilon}] = \frac{P[R_1 > \varepsilon]}{E[1-z^{R_1}; R_1 \leq \varepsilon] + P[R_1 > \varepsilon]}. \tag{4.2}$$

It can also be written in the following form

$$E[z^{L_\varepsilon}] = \frac{1-\Phi_\varepsilon(1)}{1-\Phi_\varepsilon(z)}, \tag{4.3}$$

where $\Phi_\varepsilon(z)$ is given by

$$\Phi_\varepsilon(z) = \sum_{k=0}^{(\varepsilon'-1)/2} \varphi_{1,2k+1} z^{2k+1}, \quad \varphi_{r,n} = \frac{r}{n}\binom{n}{\frac{n+r}{2}} 2^{-n}. \tag{4.4}$$

Then, the expectation and the variance of the simplified avalanche length are derived and established in Corollary 2.4.

Finally, Theorem 2.1 states that the scaled simplified avalanche length for the simple symmetric random walk converges in distribution to the simplified Brownian avalanche length, i.e. analytically we have

$$\lim_{n\to\infty} E[e^{-\frac{\lambda}{n}L_{n\varepsilon}}] = \frac{1}{\sqrt{\lambda\varepsilon\pi}\,\mathrm{erf}(\sqrt{\lambda\varepsilon}) + e^{-\lambda\varepsilon}}. \tag{4.5}$$

After establishing results for the simplified avalanche length, special focus is on the full avalanche length. In order to derive the probability generating function for the full avalanche length, firstly three sets of random walk paths are defined, namely $\mathscr{A}_\mu$, $\mathscr{B}_\mu$ and $\mathscr{C}_\mu$, and further their probability generating function $A_\mu(z)$, $B_\mu(z)$ and $C_\mu(z)$ are established in Lemma 2.6. After having those generating functions derived, the path decomposition of the path that corresponds to the Type II trade is considered. Precisely, if the first trade is a Type II trade its path can be decomposed as the concatenation of K elements from $\mathscr{B}_\mu$ and one element from $\mathscr{C}_\mu$, $K \geq 1$, as proved in Proposition 2.3. Therefore, the probability generating function of the time to the next trade $T_\mu(z)$ is given by

$$T_\mu(z) = E[z^{T_1}] = A_\mu(z) + \frac{B_\mu(z)}{1 - B_\mu(z)} \cdot C_\mu(z). \tag{4.6}$$

After having the law of T_1 established in (2.79), the probability generating function of the full avalanche length for the symmetric random walk can be computed, and it is

$$E[z^{L^*_{\mu\varepsilon}}] = \frac{P[T_1 > \varepsilon]}{E[1 - z^{T_1}; T_1 \leq \varepsilon] + P[T_1 > \varepsilon]}. \tag{4.7}$$

Furthermore, the probability generating functions for the different quantities, such as: the time to the first trade assuming it is a Type I trade, the time to the first trade assuming it is a Type II trade, and the time to the first Type II trade, are presented in Lemma 2.84. As a consequence, the probability that the first trade is a Type I trade is stated in Corollary 2.6, and it is

$$P[S(T_1) > 0] = \frac{\mu}{\mu + 1}. \tag{4.8}$$

The probabilities that the first trade, either a Type I or a Type II trade, occurs at time $n = 1, \ldots, 10$ are established in Table 2.1. Finally, for $\varepsilon = 1, 2, 3, 4, 5$ the probabilities that the full avalanche length takes values $k = 1, \ldots, 8$ are presented in Table 2.4, Table 2.5 and Table 2.6 for $\mu = 1$, $\mu = 2$ and $\mu = 3, 4$ respectively.

Section 2.6 is devoted to introduce a binomial limit order book model in which the mid-price converges in distribution to a subordinated Wiener process. Therefore, one of the direction of the possible future work is to extend the model introduce in the Section 2.6 and to incorporate order cancellations. Furthermore, the special interest for the future work would be to consider the model in which the orders are placed in the order book with respect to the Hawkes process. In addition, one of the extension can be to assume

that new limit orders are placed at every level $S_t + u$ with volume density $g(u)du$, where g is an integrable function.

Chapter 3 studies the performance of the proposed Gated Recurrent Unit network model to classify market data vector into the label (buy/idle). Firstly, in order to efficiently analyze a large amount of data, a platform that is compatible with the NASDAQ's limit order book data is proposed in Section 3.3, see Figure 3.1. Note that the proposed platform significantly speeds up the process of data transformation (see Table 3.1), and thus enables the running of various algorithms that perform feature extraction from the limit order book and transform the data into a format suitable for further research. The proposed platform contributes to working with the LOBSTER market data efficiently and effectively. Additionally, it can be used for further research projects.

Then, in order to extract particular features that describe market behavior, the data transformation part presented in Section 3.7 is utilized. Note that the technical indicators present in a library *TA-Lib* are also included during the data transformation. One of the contributions of this thesis to the field of machine learning applications is the implementation of the GRU based neural network to deal with the stock market data. In order to be able to run the GRU network model, a label is assigned to each vector of the training set by employing Algorithm 3.7.1.

After incorporating also the Fourier transformation based features, different approaches to perform feature selection are proposed in Section 3.9.5 and they represent another contribution of this thesis. To perform a feature selection, two assumptions are made. Firstly, the higher absolute value of the feature autocorrelation the higher likely-hood of the indicator imposing repetitive pattern and thus knowledge can be inferred easier. Secondly, a feature having a higher value of mutual information with the label holds more information relevant for classifying the market data vector into the label. After a *SCORE* value is assigned to each feature, the Stochastic universal sampling (SUS) is implemented in order to select features with respect to their score (higher the score, higher the likelihood for the feature to be selected). The performance of the model fed with the features selected by proposed methods is improved when compared to the same model using the randomly selected features.

In addition, the features utilizing Fourier Transformation offer statistically significant improvement of the GRU model's performance. The presented statistical analysis showed that the main assumption is confirmed with a significance level of 0.01, see Table 3.5. This finding shows that observing the stock market price as a signal and employing frequen-

cy decomposition techniques leads to strong results and brings a statistically significant improvement into the decision making model. The next observation in this thesis is that the mutual-information, when combined with the SUS algorithm outperforms the random selection of features with a statistical significance of 0.05. Regarding the autocorrelation based feature ranking, even though the average observed *F1* score has improved, this score ranking method has not shown significant results (for statistical significance $\alpha = 0.05$). To conclude, the proposed methods were beneficial for selecting adequate features from large set of market data features and these methods could significantly contribute to various future research projects that observe the market data.

Future work will be continued in several directions. Since exploring the influence of social networks on a stock market can be beneficial, a possible enhancement of the presented research is the incorporation of the social network data, e.g. twitter data. Moreover, future studies could incorporate the wavelet transform, which could be very informative, in order to extract new features. Besides, further research is needed to investigate the performance of different models that can be used instead of the GRU model, e.g. the RNN and LSTM, and compare the results of all three.

5

Appendix

5.1 The code that reproduces Figure 2.2

```
/* This code generates TikZ commands for Figure 2.2. that
   illustrates a Type II trade, that is followed by two consecutive
    Type~I trades*/

#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<string.h>

int main()
{
/*
 N...the length of the random walk
 Z...string with up '+' and down '-'
 X...increments
 S...random walk
 M...running maximum
 rh...simplified trading times
 R...simplified intertrading times

*/
  FILE *fp;
  int *X,*S,*A,*B,*rh,*R;
  int N,n,K,k,i;
```

```
23  int  mu=2; /*Set a displacemnt integer to the adequate value. */
24
25
26  /* (+) for step up and (-) for step down in the simple symmetric
     random walk; */
27  char Z[]={"0---++++"};
28  int eps = 7; /*Set epsilon to the adequate value. */
29
30  /* Length of the random walk */
31
32  N=strlen(Z)-1;
33
34  /* Allocate an array for increments, the random walk, the maximum
      and the ladder times */
35  X=(int *)calloc(N+1,sizeof(int));
36  S=(int *)calloc(N+1,sizeof(int));
37  A=(int *)calloc(N+1,sizeof(int));
38  B=(int *)calloc(N+1,sizeof(int));
39  rh=(int *)calloc(N+1,sizeof(int));
40  R=(int *)calloc(N+1,sizeof(int));
41
42
43  /* initial values for random walk, minimum, maximum */
44  S[0]=0;
45  A[0]=0;
46  B[0]=0;
47
48  /* initial ladder time */
49  rh[0]=0;
50  K=0;
51
52  printf("%d %d %d\n",0,S[0],B[0]);
53
54  for(n=1;n<=N;n++)
55  {
56    /* Compute increments from string, if '+' take +1, otherwise -1
      */
57    X[n]=Z[n]=='+'?1:-1;
58
59    /* Compute random walk */
```

```
60      S[n]=S[n-1]+X[n];
61
62      /* Compute minimum */
63      if(S[n]<A[n-1])
64      {
65        A[n]=S[n];
66      }
67      else
68      {
69        A[n]=A[n-1];
70      }
71
72      /* Compute maximum and ladder times */
73      if(S[n]>B[n-1])
74      {
75        B[n]=S[n];
76        K++;
77        rh[K]=n;
78        R[K]=rh[K]-rh[K-1];
79      }
80      else
81      {
82         B[n]=B[n-1];
83      }
84      printf("%d %d %d\n",n,S[n],B[n]);
85    }
86
87    printf("Ladder times\n");
88    for(k=1;k<=K;k++)
89    {
90      printf("%d %d %d\n",k,rh[k],R[k]);
91    }
92    /*Volume process allocate space*/
93    int** allocateMatrix(int n, int m){
94    int i;
95    int ** V = malloc(n*sizeof(int*));
96    if(V == NULL)
97    {
98      fprintf(stderr, "-1\n");
99      exit(EXIT_FAILURE);
```

```
  }

  for(i=0; i<n; i++)
  {
    V[i] = malloc(m*sizeof(int));
    if(V[i] == NULL)
    {
      fprintf(stderr, "-1\n");
      exit(EXIT_FAILURE);
    }
  }
  return V;
}

  int umax,umin;
  umin=A[N]-1;
  umax=B[N]+mu+1;
 int M;
 M=umax-umin + 1;
 int **V;
  /*volume process introduce dynamics*/
  V = allocateMatrix(N+1, M);

  printf("umin = %d, umax = %d, M= %d\n", umin, umax, M);
                //umin is negative number
  int j;          //u ->  umin umin+1 umin+2 ...    0       ... umax
  for(j=0; j<M; j++){   //j ->  0   1    ....   -umin   ... M-1
  if(j+umin>=0)
    V[0][j] = 1;
  else
    V[0][j] = 0;
  }

  for(n=1; n<=N; n++){
    for(j=0; j<M; j++){
      V[n][j] = V[n-1][j];
      V[n][S[n-1]-umin+mu]=V[n-1][S[n-1]-umin+mu]+1;
      V[n][S[n-1]-umin] = 0;
    }
  }
```

```
140
141  for(j=M-1; j>=0; j--){
142    for(n=0; n<=N; n++){
143      printf("%d ", V[n][j]);
144    }
145    printf("\n");
146  }
147   /* Trading times tau and last trading time xi */
148  int *ta = (int *)calloc(N+1, sizeof(int));
149  int *xi = (int *)calloc(N+1, sizeof(int));
150  int nta = 0;
151  ta[0] = 0;
152  xi[0] = 0;
153  for(n=0; n<=N; n++){
154    xi[n] = ta[nta];
155    printf("n = %d  S[n] = %d  V[n][S[n]] = %d\n", n, S[n], V[n][S[
     n]-umin]);
156    if(V[n][S[n]-umin]>0){
157      nta = nta +1;
158      ta[nta] = n;
159    }
160  }
161  printf("nta = %d\n",nta);
162  /* Trades of Type 1 and 2 */
163  int *ta1 = (int *)calloc(N+1, sizeof(int));
164  int *ta2 = (int *)calloc(N+1, sizeof(int));
165  int nta1 = 0, nta2 = 0;
166  ta1[0] = -1;
167  ta2[0] = 0;
168  for (i=1; i<=nta; i++){
169    if(S[ta[i]]>S[ta[i-1]]){
170      nta1 = nta1+1;
171      ta1[nta1] = ta[i];
172    }
173    else{
174      nta2 = nta2 + 1;
175      ta2[nta2] = ta[i];
176    }
177  }
178
```

```
179   /* open the TeX file for writing */
180   fp=fopen("figType2.tex","w");
181
182   /* print documentclass command */
183   fprintf(fp,"\\documentclass{standalone}\n");
184
185   /* print usepackage command */
186   fprintf(fp,"\\usepackage{tikz}\n");
187
188   /* print TikZ library command */
189   fprintf(fp,"\\usetikzlibrary{shapes,patterns,decorations.
       pathreplacing}\n");
190
191   /* print begin document command */
192   fprintf(fp,"\\begin{document}\n");
193
194   /* print begin TikZ picture command */
195   fprintf(fp,"\\begin{tikzpicture}[x={(10mm,0mm)},y={(0mm,10mm)}]\n
       ");
196
197   /* print commands for drawing the random walk with lines */
198   fprintf(fp,"\\draw [very thick] plot coordinates {");
199   for(n=0;n<=N;n++)
200   {
201     fprintf(fp,"(%d,%d) ",n,S[n]);
202     if(n%10==0) fprintf(fp,"\n");
203   }
204   fprintf(fp,"};\n");
205
206   /* print commands for axes */
207   fprintf(fp,"\\draw[->] (-1,0) -- (%d,0);\n",N+1);
208   fprintf(fp,"\\draw[->] (0,%d) -- (0,%d);\n",A[N]-2,B[N]+5);
209
210 /* print command for axes labels */
211   fprintf(fp,"\\draw (0,%d) node[left=1pt] {$S_n$};\n",B[N]+5);
212   fprintf(fp,"\\draw (%d,0) node[below=1pt] {$n$};\n",N+1);
213
214   fprintf(fp,"\\tikzset{\n");
215   fprintf(fp,"  lp/.style={circle,draw,thick,inner sep=0pt,minimum
       size=3mm},\n");
```

```
216  fprintf(fp,"  op/.style={circle,fill,inner sep=0pt,minimum size
      =1.5mm},\n");
217  fprintf(fp,"  vol/.style={rectangle,draw=black!50,thick,inner sep
      =0pt,minimum size=4mm},\n");
218  fprintf(fp,"  ta1/.style={rectangle,draw=blue,yscale=1.2,rotate
      =45,thick,inner sep=0pt,minimum size=4mm},rounded corners=3pt,\n
      ");
219  fprintf(fp,"  ta2/.style={rectangle,draw=red,thick,inner sep=0pt,
      minimum size=4mm}\n");
220  fprintf(fp,"}\n");
221
222  /* print volume rectangles */
223  for(n=0; n<=N; n++){
224    fprintf(fp,"\\path ");
225    for(j=0; j<M; j++){
226      if(V[n][j]>0){
227            fprintf(fp,"(%d, %d) node[vol]{}\n", n, j+umin);
228      }
229    }
230  fprintf(fp,";\n");
231  }
232
233  /* print commands for drawing the random walk with points */
234  for(n=0;n<=N;n++)
235  {
236    fprintf(fp,"  \\path (%d,%d) node [op] {};\n",n,S[n]);
237    if(n%10==0) fprintf(fp,"\n");
238  }
239
240  /* type I */
241  printf("nta1 = %d \n", nta1);
242  for(k=1; k<=nta1; k++){
243      printf("  \\path (%d,%d) node [ta1] {};\n",ta1[k],S[ta1[k]]);
244
245        fprintf(fp,"  \\path (%d,%d) node [ta1] {};\n",ta1[k],S[ta1
       [k]]);
246  }
247
248  /* type II */
249  printf("nta2 = %d \n", nta2);
```

```
  for(k=1; k<=nta2; k++){
      printf("  \\path (%d,%d) node [ta2] {};\n",ta2[k],S[ta2[k]]);

        fprintf(fp,"  \\path (%d,%d) node [ta2] {};\n",ta2[k],S[ta2
   [k]]);
  }

  /* print end TikZ picture command */
  fprintf(fp,"\\end{tikzpicture}\n");

  /* print end document command */
  fprintf(fp,"\\end{document}\n");

  /* Free the arrays*/
  free(R);
  free(rh);
  free(B);
  free(A);
  free(S);
  free(X);
   return 0;
}
```

5.2 The code that reproduces Figure 2.6

```
/* This code generates TikZ commands for Figure 2.6 */

#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<string.h>

int main()
{
/*
 N...the length of the random walk
 Z...string with up '+' and down '-'
 X...increments
 S...random walk
 M...running maximum
```

```
rh...simplified
R...simplified inter
*/
 FILE *fp;
 int *X,*S,*A,*B,*rh,*R;
 int N,n,K,k;
 int mu=2;

 /* + denotes the step up, while - denotes the down step of the
  simple symmetric random walk */
 char Z[]={"0+--+++-++-+---+--++"};
 int eps = 7; /*Set epsilon to the adequate value. */

 /* Length of the random walk */
  N=strlen(Z)-1;

 /* Allocate an array for increments, the random walk, the maximum
  , and the ladder times */
 X=(int *)calloc(N+1,sizeof(int));
 S=(int *)calloc(N+1,sizeof(int));
 A=(int *)calloc(N+1,sizeof(int));
 B=(int *)calloc(N+1,sizeof(int));
 rh=(int *)calloc(N+1,sizeof(int));
 R=(int *)calloc(N+1,sizeof(int));

 /* initial values for random walk, minimum, maximum */
 S[0]=0;
 A[0]=0;
 B[0]=0;

 /* initial ladder time */
 rh[0]=0;
 K=0;

 printf("%d %d %d\n",0,S[0],B[0]);

 for(n=1;n<=N;n++)
 {
   /* Compute increments from string, if '+' take +1, otherwise -1
   */
```

```
53      X[n]=Z[n]=='+'?1:-1;
54
55
56      /* Compute random walk */
57      S[n]=S[n-1]+X[n];
58
59      /* Compute minimum */
60      if(S[n]<A[n-1])
61      {
62        A[n]=S[n];
63      }
64      else
65      {
66        A[n]=A[n-1];
67      }
68
69      /* Compute maximum and ladder times */
70      if(S[n]>B[n-1])
71      {
72        B[n]=S[n];
73        K++;
74        rh[K]=n;
75        R[K]=rh[K]-rh[K-1];
76      }
77      else
78      {
79         B[n]=B[n-1];
80      }
81      printf("%d %d %d\n",n,S[n],B[n]);
82    }
83
84    printf("Ladder times\n");
85    for(k=1;k<=K;k++)
86    {
87      printf("%d %d %d\n",k,rh[k],R[k]);
88    }
89    /*Volume process allocate space*/
90    int** allocateMatrix(int n, int m){
91    int i;
92    int ** V = malloc(n*sizeof(int*));
```

```
93    if(V == NULL)
94    {
95      fprintf(stderr, "-1\n");
96      exit(EXIT_FAILURE);
97    }
98
99    for(i=0; i<n; i++)
100   {
101     V[i] = malloc(m*sizeof(int));
102     if(V[i] == NULL)
103     {
104       fprintf(stderr, "-1\n");
105       exit(EXIT_FAILURE);
106     }
107   }
108   return V;
109 }
110
111   int umax,umin;
112   umin=A[N]-1;
113   umax=B[N]+mu+1;
114  int M;
115  M=umax-umin + 1;
116  int **V;
117   /*volume process introduce dynamics*/
118   V = allocateMatrix(N+1, M);
119
120   printf("umin = %d, umax = %d, M= %d\n", umin, umax, M);
121                 //umin is negative number
122   int j;          //u ->  umin umin+1 umin+2 ...    0       ... umax
123   for(j=0; j<M; j++){   //j ->  0    1    ....   -umin   ... M-1
124   if(j+umin>=0)
125     V[0][j] = 1;
126   else
127     V[0][j] = 0;
128   }
129
130   for(n=1; n<=N; n++){
131     for(j=0; j<M; j++){
132       V[n][j] = V[n-1][j];
```

```
133       V[n][S[n]-umin+mu]=V[n-1][S[n]-umin+mu]+1;
134       V[n][S[n-1]-umin] = 0;
135     }
136   }
137
138   for(j=M-1; j>=0; j--){
139     for(n=0; n<=N; n++){
140       printf("%d ", V[n][j]);
141     }
142     printf("\n");
143   }
144      /* Trading times tau and last trading time xi */
145   int *ta = (int *)calloc(N+1, sizeof(int));
146   int *xi = (int *)calloc(N+1, sizeof(int));
147   int nta = 0;
148   ta[0] = 0;
149   xi[0] = 0;
150   for(n=0; n<=N; n++){
151     xi[n] = ta[nta];
152     printf("n = %d  S[n] = %d  V[n][S[n]] = %d\n", n, S[n], V[n][S[
      n]-umin]);
153     if(V[n][S[n]-umin]>0){
154       nta = nta +1;
155       ta[nta] = n;
156     }
157   }
158   printf("nta = %d\n",nta);
159   /* Trades of Type 1 and 2 */
160   int *ta1 = (int *)calloc(N+1, sizeof(int));
161   int *ta2 = (int *)calloc(N+1, sizeof(int));
162   int i, nta1 = 0, nta2 = 0;
163   ta1[0] = -1;
164   ta2[0] = 0;
165   for (i=1; i<=nta; i++){
166     if(S[ta[i]]>S[ta[i-1]]){
167       nta1 = nta1+1;
168       ta1[nta1] = ta[i];
169     }
170     else{
171       nta2 = nta2 + 1;
```

```
172       ta2[nta2] = ta[i];
173     }
174   }
175
176   /* open the TeX file for writing */
177   fp=fopen("fig_simplifiedAvalanche.tex","w");
178
179   /* print documentclass command */
180   fprintf(fp,"\\documentclass{standalone}\n");
181
182   /* print usepackage command */
183   fprintf(fp,"\\usepackage{tikz}\n");
184
185   /* print TikZ library command */
186   fprintf(fp,"\\usetikzlibrary{shapes,patterns,decorations.
       pathreplacing}\n");
187
188   /* print begin document command */
189   fprintf(fp,"\\begin{document}\n");
190
191   /* print begin TikZ picture command */
192   fprintf(fp,"\\begin{tikzpicture}[x={(10mm,0mm)},y={(0mm,10mm)}]\n
       ");
193
194   /* print commands for drawing the random walk with lines */
195   fprintf(fp,"\\draw [very thick] plot coordinates {");
196   for(n=0;n<=N;n++)
197   {
198     fprintf(fp,"(%d,%d) ",n,S[n]);
199     if(n%10==0) fprintf(fp,"\n");
200   }
201   fprintf(fp,"};\n");
202
203   /* print commands for axes */
204   fprintf(fp,"\\draw[->] (-1,0) -- (%d,0);\n",N+1);
205   fprintf(fp,"\\draw[->] (0,%d) -- (0,%d);\n",A[N]-2,B[N]+5);
206
207   /* print command for axes labels */
208   fprintf(fp,"\\draw (0,%d) node[left=1pt] {$S_n$};\n",B[N]+5);
209   fprintf(fp,"\\draw (%d,0) node[below=1pt] {$n$};\n",N+1);
```

```
210
211 /* define a style called 'lp' for a ladder point and 'op' for
      ordinary point*/
212   fprintf(fp,"\\tikzset{\n");
213   fprintf(fp,"  lp/.style={circle,draw,thick,inner sep=0pt,minimum
       size=3mm},\n");
214   fprintf(fp,"  op/.style={circle,fill,inner sep=0pt,minimum size
       =1.5mm},\n");
215    fprintf(fp,"  alpha/.style={circle,fill=red,inner sep=0pt,
       minimum size=1.5mm},\n");
216   fprintf(fp,"  old/.style={rectangle,draw=black!50,thick,inner sep
       =0pt,minimum size=4mm},\n");
217   fprintf(fp,"  ta1/.style={rectangle,draw=blue,yscale=1.2,rotate
       =45,thick,inner sep=0pt,minimum size=4mm},rounded corners=3pt,\n
       ");
218   fprintf(fp,"  ta2/.style={rectangle,draw=red,thick,inner sep=0pt,
       minimum size=4mm}\n");
219   fprintf(fp,"}\n");
220
221
222
223   /* print volume rectangles */
224   for(n=0; n<=N; n++){
225     fprintf(fp,"\\path ");
226     for(j=0; j<M; j++){
227       if(V[n][j]>0){
228             fprintf(fp,"(%d, %d) node[old]{}\n", n, j+umin);
229       }
230     }
231   fprintf(fp,";\n");
232   }
233
234   /* print commands for drawing the random walk with points */
235   for(n=0;n<=N;n++)
236   {
237     fprintf(fp,"  \\path (%d,%d) node [op] {};\n",n,S[n]);
238     if(n%10==0) fprintf(fp,"\n");
239   }
240
241   /* print commands for ladder points marked by circles */
```

```
242   for(k=0;k<=K;k++)
243   {
244     fprintf(fp,"  \\path (%d,%d) node [lp] {};\n",rh[k],k);
245     fprintf(fp,"\\draw[-] (%d,%d) -- (%d,%d);\n",rh[k],A[N]-1,rh[k
      ],B[N]+1);
246 }
247
248   fprintf(fp,"\\draw[dashed] (%d,%d) -- (%d,%d);\n",rh[K]+eps,A[N
      ]-1,rh[K]+eps,B[N]+1);
249
250   /* L Line */
251   fprintf(fp,"\\draw[-, red] (%d,%.1f) -- (%d,%.1f) node [red,
      midway,anchor=north,yshift=13pt] {$L_{\\varepsilon}$};\n",0,A[N
      ]-0.5,rh[K],A[N]-0.5);
252
253   for(k=1;k<=K;k++)
254   {
255     fprintf(fp,"\\draw [decorate,decoration={brace,amplitude=10pt,
      mirror}](%d,%d) -- (%d,%d) node [black,midway,anchor=north,
      yshift=-10pt] {$R_%d=%d$};\n",rh[k-1],A[N]-1,rh[k],A[N]-1,k,R[k
      ]);
256
257   }
258
259    fprintf(fp,"\\draw [red,decorate,decoration={brace,amplitude=10
      pt,mirror}](%d,%d) -- (%d,%d) node [red,midway,anchor=north,
      yshift=-10pt] {$R_%d>\\varepsilon$};\n",rh[K],A[N]-1,rh[K]+eps,A
      [N]-1,k);
260
261   /* type I trades*/
262   printf("nta1 = %d \n", nta1);
263   for(k=1; k<=nta1; k++){
264       printf("  \\path (%d,%d) node [ta1] {};\n",ta1[k],S[ta1[k]]);
265
266         fprintf(fp,"  \\path (%d,%d) node [ta1] {};\n",ta1[k],S[ta1
      [k]]);
267   }
268
269   /* type II trades*/
270   printf("nta2 = %d \n", nta2);
```

```
271   for(k=1; k<=nta2; k++){
272       printf("  \\path (%d,%d) node [ta2] {};\n",ta2[k],S[ta2[k]]);
273         fprintf(fp,"  \\path (%d,%d) node [ta2] {};\n",ta2[k],S[ta2
      [k]]);
274   }
275
276   /* print end TikZ picture command */
277   fprintf(fp,"\\end{tikzpicture}\n");
278
279   /* print end document command */
280   fprintf(fp,"\\end{document}\n");
281
282   /* Free the arrays*/
283   free(R);
284   free(rh);
285   free(B);
286   free(A);
287   free(S);
288   free(X);
289
290   return 0;
291 }
```

5.3 The number of paths

5.3.1 The number of paths corresponding to the first simplified trade

By the binomial formula, we have

$$\sqrt{1-z^2} = \sum_{n\geq 0} \binom{\frac{1}{2}}{n} (-1)^n z^{2n}, \tag{5.1}$$

and then the following holds

$$\frac{1-\sqrt{1-z^2}}{z} = \sum_{n\geq 1} \binom{\frac{1}{2}}{n} (-1)^{n-1} z^{2n-1}. \tag{5.2}$$

Furthermore, we have

$$\begin{aligned}\binom{\frac{1}{2}}{n} &= \frac{\frac{1}{2}(\frac{1}{2}-1)\cdots(\frac{1}{2}-n+1)}{n!} = \frac{1}{2^n}\frac{1(1-2)(1-4)\cdots(1-2n+2)}{n!} \\ &= \frac{(-1)^{n-1}}{2^n}\frac{1\cdot 3\cdots(2n-3)}{n!}\frac{2\cdot 4\cdots(2n-2)}{2\cdot 4\cdots(2n-2)} \\ &= \frac{(-1)^{n-1}}{2^{2n-1}}\frac{(2n-2)!}{n!(n-1)!} = \frac{1}{2n-1}\frac{(-1)^{n-1}}{2^{2n-1}}\binom{2n-1}{n}.\end{aligned} \tag{5.3}$$

Thus, the following equality holds

$$\frac{1-\sqrt{1-z^2}}{z} = \sum_{n\geq 1}\frac{1}{2n-1}\frac{1}{2^{2n-1}}\binom{2n-1}{n}z^{2n-1}. \tag{5.4}$$

By substituting $k = 2n-1$, the corresponding formula for a_k follows from the probability generating function, which is obtained from the ordinary generating function by the substitution $z \mapsto z/2$.

5.3.2 The number of paths corresponding to the trades that occur at simplified trading times

n \ r	1	2	3	4	5	6	7	8	9	10
1	1									
2	0	1								
3	1	0	1							
4	0	2	0	1						
5	2	0	3	0	1					
6	0	5	0	4	0	1				
7	5	0	9	0	5	0	1			
8	0	14	0	14	0	6	0	1		
9	14	0	28	0	20	0	7	0	1	
10	0	42	0	48	0	27	0	8	0	1

Table 5.1: *The number of random walk paths of length $n = 1,\ldots,10$ that lead to trades that occur at price level $r = 1,\ldots,n$ for the first time*

Since an order book is initially full, the paths that lead to trades that occur at price

level r for the first time are paths of simple random walk paths with first passage through r. Below you can find a code written in programming language R that contains three methods computing the number of those paths of finite length $n = 1, \ldots, 10$. For particular values see Table 5.1.

```
# The number of simple symmetric random walk paths
# with first passage through r=1,...,n at step n=1,...,10
install.packages("elliptic")
install.packages("binaryLogic")
N<- 10
# Method 1: Explicit formula
for(n in 1:N){
  P<- rep(0,n)
  for(r in 1:n){
    P[r]<- ifelse((n+r)%%2==0,r/n*choose(n,(n+r)%/%2),0)
  }
  print(P)
}
# Method 2: Cauchy formula and generating function
require(elliptic)
for(n in 1:N){
  P<- rep(0,n)
  for(r in 1:n){
    fun<- function(z)((1-sqrt(1-4*z^2))/(2*z))^r/z^n
    P[r]<- round(Re(residue(fun,0,0.5)))
  }
  print(P)
}
# Method 3: Brute fource counting
require(binaryLogic)
for(n in 1:N){
  P<- rep(0,n)
  for(w in 0:(2^n-1)){
    X<- 2*as.integer(as.vector(as.binary(w,n=n)))-1
    S<- diffinv(X)
    for(r in 1:n){
      if(S[n+1]==r&&max(S[1:n])<r) P[r]<- P[r]+1
    }
  }
  print(P)
}
```

μ \ n	1	2	3	4	5	6	7	8	9	10	11
1	1										
2	0	0									
3	0	1	1								
4	0	0	0	0							
5	0	1	2	2	2						
6	0	0	0	0	0	0					
7	0	1	4	5	5	5	5				
8	0	0	0	0	0	0	0	0			
9	0	1	8	13	14	14	14	14	14		
10	0	0	0	0	0	0	0	0	0	0	
11	0	1	16	34	41	42	42	42	42	42	42

Table 5.2: *The number of paths of length n from the class* $\mathscr{A}_\mu$. *The illustration is with* $n = 1,...,10,11$ *and* $\mu = 1,...,n$

5.3.3 The number of paths that belong to the class $\mathscr{A}_\mu$

Table 5.3.3 present the number of paths of finite length $n = 1,...,10,11$ that belong to the class $\mathscr{A}_\mu$, when $\mu = 1,...,n$. Below you can find a code written in programming language *R* that contains three methods computing the number of paths of finite length $n = 1,...,10,11$ from the class $\mathscr{A}_\mu$. For particular values see Table 5.3.3. Note that the number of paths that belongs to the class $\mathscr{B}_\mu$ and $\mathscr{C}_\mu$ can be determined in the same manner.

```
# Method 1: Cauchy formula and generating function
require(elliptic)

l1<- function(z)((1+sqrt(1-z^2))/z)
l2<- function(z)((1-sqrt(1-z^2))/z)
#for different values for m
N=15
A<- function(m,z)(l1(z)^m-l2(z)^m)/(l1(z)^(m+1)-l2(z)^(m+1))

for(n in 1:N){
  P<- rep(0,n)
  r<-1
  for(m in 1:n){
```

```
#     fun<- function(z)(((((1+sqrt(1-z^2))/z)^m-((1-sqrt(1-z^2))/z)^
    m)/(((1+sqrt(1-z^2))/z)^(m+1))-((1-sqrt(1-z^2))/z)^(m+1)))^r/z^n
     fun <-function(z) A(m,2*z)/z^n
     P[m]<- round(Re(residue(fun,0,0.5)))
  }
  print(P)
}
# Method 2: Brute fource counting
require(binaryLogic)
for(n in 1:N){
  P<- rep(0,n)
  for(w in 0:(2^n-1)){
    X<- 2*as.integer(as.vector(as.binary(w,n=n)))-1
    S<- diffinv(X)
    for(m in 1:n){
      if(S[n+1]==1&&max(S[1:n])<=0 &&min(S[1:n]>-m)) P[m]<- P[m]+1
    }
  }
  print(P)
}
```

On convergence in distribution and vague convergence

Lemma 5.1. *Let $\varepsilon > 0$ be a real number $\varepsilon > 0$, and suppose we are given a sequence of probability measures $(\mu_n)_{n\geq 1}$ on $\mathscr{B}((0,\infty))$ and a non-negative , not the zero measure ν on $\mathscr{B}((0,\infty))$ such that*

$$\int_0^\infty (1\wedge x)\nu(dx) < \infty. \tag{5.5}$$

If ε is a continuity point for ν and

$$\int_0^\infty e^{-\lambda x}\mu_n(dx) = 1 - \hat{\nu}(\lambda)\cdot n^{-1/2} + \mathscr{O}(n^{-1}) \quad n\to\infty \tag{5.6}$$

pointwise for all $\lambda \geq 0$, where

$$\hat{\nu}(\lambda) = \int_0^\infty (1-e^{-\lambda x})\nu(dx), \tag{5.7}$$

then we have

$$\lim_{n\to\infty} n^{1/2}\int_0^\varepsilon (1-e^{-\lambda x})\mu_n(dx) = \int_0^\varepsilon (1-e^{-\lambda x})\nu(dx) \tag{5.8}$$

and

$$\lim_{n\to\infty} n^{1/2}\int_\varepsilon^\infty \mu_n(dx) = \int_\varepsilon^\infty \nu(dx) \tag{5.9}$$

for all $\lambda \geq 0$.

Proof. Assumption (5.5) implies that the integral in (5.7) is finite for all $\lambda \geq 0$. Let us define $\tilde{\mu}_n(\lambda)$ by

$$\tilde{\mu}_n(\lambda) = \int_0^\varepsilon (1-e^{-\lambda x})\mu_n(dx),$$

which is the integral on the left hand side of (5.8), and it is the Laplace transform of μ_n. The case $\lambda = 0$ is trivial, so fix $\lambda > 0$.

Since we assumed that μ_n is defined on $\mathscr{B}((0,\infty))$ we have $0 \leq \tilde{\mu}_n(\lambda) < 1$ for $\lambda > 0$.

Let us define measures γ_n and $\tilde{\gamma}_n$ by $\gamma_n(dx) = n^{1/2}\mu_n(dx)$ and $\tilde{\gamma}_n(dx) = n^{1/2}\tilde{\mu}_n(dx)$.

Furthermore, since ν is not the zero measure, we have $\hat{\nu}(\lambda) > 0$. Let us define another new measure ν_λ by

$$\nu_\lambda(dx) = \frac{1-e^{-\lambda x}}{\hat{\nu}(\lambda)}\mu_n(dx). \tag{5.10}$$

Note that ν_λ is a probability measure. Denote by $\hat{\nu}_\lambda(s)$ its Laplace transform.

The asymptotics (5.6) imply

$$\lim_{n\to\infty} \tilde{\gamma}_n(\lambda) = \hat{\nu}_\lambda(\lambda) \tag{5.11}$$

pointwise for all $s \geq 0$.

By the continuity theorem for Laplace transforms, e.g. [52, Theorem XIII.1.2a, P.433], it follows that $\gamma_n \to \nu_\lambda$ weakly as $n \to \infty$. As ε is also a continuity point for the limit distribution ν_λ, it follows

$$\lim_{n\to\infty} \int_0^\varepsilon \gamma_n(dx) = \int_0^\varepsilon \nu_\lambda(dx). \tag{5.12}$$

Relation (5.6) implies also

$$\lim_{n\to\infty} n^{1/2}(1-\tilde{\mu}_n(\lambda)) = \hat{\nu}(\lambda). \tag{5.13}$$

Combining (5.12) and (5.13) yields (5.9). Now consider the function

$$f(x) = \frac{\hat{\nu}(\lambda)}{1-e^{-\lambda x}} I_{x>\varepsilon}, \quad x > 0. \tag{5.14}$$

It is bounded and continuous except for the point $x = \varepsilon$. By the continuous mapping theorem, as formulated for example in [47, Theorem 3.2.4, P.101] we get

$$\lim_{n\to\infty} \int_\varepsilon^\infty f(x)\gamma_n(dx) = \int_\varepsilon^\infty f(x)\nu_\lambda(dx). \tag{5.15}$$

Using the definitions of γ_n and ν_λ and (5.13) we obtain (5.8). □

Proposition 5.1. *In the setting of Lemma 5.1 we have*

$$\lim_{n\to\infty} n^{1/2}\mu_n = \nu \tag{5.16}$$

vaguely on $(0,\infty)$.

Proof. Suppose f is a continuous function with compact support in $(0,\infty)$. Define $g(x)$ for $x > 0$ as $g(x) = \hat{\nu}(\lambda)f(x)/(1-e^{-\lambda x})$. Since f vanishes in some neighborhood of $x = 0$, it follows that g is a bounded continuous function. Thus we have from weak convergence

$$\lim_{n\to\infty} \int_0^\infty g(x)\gamma_n(dx) = \int_0^\infty g(x)\nu(dx), \tag{5.17}$$

which can be rewritten with (5.13) as

$$\lim_{n\to\infty} \int_0^\infty f(x) n^{1/2} \mu_n(dx) = \int_0^\infty f(x) \nu(dx), \tag{5.18}$$

thus showing vague convergence. □

5.4 Auxiliary lemmas

Lemma 5.2. *Assume that $f_n \to f$ locally uniformly, where f is a continuous function and $g_n \to g$ in $D([0,\infty])$. Then, we have*

$$f_n \circ g_n \to f \circ g \text{ in } D([0,\infty]).$$

Proof. We have

$$g_n \to g \text{ in } D([0,\infty]) \quad \Leftrightarrow \rho(g_n, g) \to 0, \text{where} \tag{5.19}$$

$$\rho(f,g) = \inf_{\lambda} \max\{\|\lambda - \mathbb{I}_d\|, \|f - g \circ \lambda\|\}. \tag{5.20}$$

Since $\rho(g_n, g) < \varepsilon/4$, there exist a sequence of increasing continuous functions $\{\lambda_n\}_{n\geq 0}$ such that

$$\|\lambda_n - \mathbb{I}_d\| < \varepsilon/4, \tag{5.21}$$

$$\text{and} \quad \|g_n - g \circ \lambda_n\| < \varepsilon/4. \tag{5.22}$$

Since $f_n \to f$ locally uniformly, then

$$\|f_n - f\| < \varepsilon/2. \tag{5.23}$$

Further $\rho(f_n \circ g_n, f \circ g) < \varepsilon/2$ means that there exist a sequence $\{\tilde{\lambda}_n\}_{n\geq 0}$ such that

$$\left\|\tilde{\lambda}_n - \mathbb{I}_d\right\| < \varepsilon/2, \quad \left\|f_n \circ g_n - f \circ g \circ \tilde{\lambda}_n\right\| < \varepsilon. \tag{5.24}$$

We have

$$\begin{aligned} \|f_n \circ g_n - f \circ g \circ \lambda_n\| &\leq \|f_n \circ g_n - f \circ g_n\| + \|f \circ g_n - f \circ g \circ \lambda_n\| \\ &\overset{(5.23)}{<} \varepsilon/2 + \|f \circ g_n - f \circ g \circ \lambda_n\| \\ &\overset{(5.22)}{<} \varepsilon/2 + \varepsilon/2, \end{aligned} \tag{5.25}$$

where the last inequality is obtained since f is continuous and (5.22) holds. □

Lemma 5.3. *For an invertible function f the following equality holds*

$$\left(\frac{f(nt)}{n^{-\alpha}}\right)^{(-1)} = \frac{f^{(-1)}(n^{-\alpha}t)}{n}.$$

Proof. Denote by $\widetilde{f} = \frac{f(nt)}{n^{-\alpha}}$ and by $\widetilde{f}^{-1} = \frac{f^{(-1)}(n^{-\alpha}t)}{n}$. Since the generalized inverse is unique it is enough to show that $\widetilde{f}^{-1}(\widetilde{f}(x)) = x$ and $\widetilde{f}(\widetilde{f}^{-1}(y)) = y$.

$$\frac{f^{(-1)}(n^{-\alpha}(\frac{f(nt)}{n^{-\alpha}}))}{n} = \frac{f^{(-1)}f(nt)}{n} = t$$

$$\frac{f(n\frac{f^{(-1)}(n^{-\alpha}t)}{n})}{n^{-\alpha}} = \frac{f(f^{-1}(n^{-\alpha}t))}{n^{-\alpha}} = t.$$

□

Lemma 5.4. *Let $\{f_n\}$ be a sequence if non-decreasing functions*

$$f_n \to f \quad n \to \infty, \text{ in } D([0,\infty]), f_n(0) = 0, f_n(+\infty) = \infty,$$

where f_0 is strictly increasing. Then $f_n^{(-1)} \to f_0^{(-1)}, n \to \infty$ locally uniformly.

Proof. Assume that the statement we want to prove is false, i.e. assume that there exist a $y_n \to y$ such that without loss of generality

$$f_n^{(-1)}(y) > f_0^{(-1)}(y) + c. \tag{5.26}$$

Then,

$$\begin{aligned} 0 =& y_n - y_n = f_n(f_n^{(-1)}(y_n)) - f_0(f_0^{(-1)}(y_n)) \\ &= (f_n(f_n^{(-1)}(y_n)) - f_0(f_n^{(-1)}(y_n))) + (f_0(f_n^{(-1)}(y_n)) - f_0(f_0^{(-1)}(y_n))) \\ &> (f_n(f_n^{(-1)}(y_n)) - f_0(f_n^{(-1)}(y_n))) + \frac{\delta}{2} \\ &> \frac{\delta}{2} + \frac{\delta}{2} = \delta \end{aligned} \tag{5.27}$$

where the first inequality holds because of the strictly monotonicity of f and (5.26). The last inequality in (5.27) holds because $f_n \to f \quad n \to \infty,$ in $D([0,\infty])$ and by continuity of f at y, and in case that y is not a point of continuity f the last inequality holds because of the right continuity.

Therefore, by contradiction the lemma is proved.

□

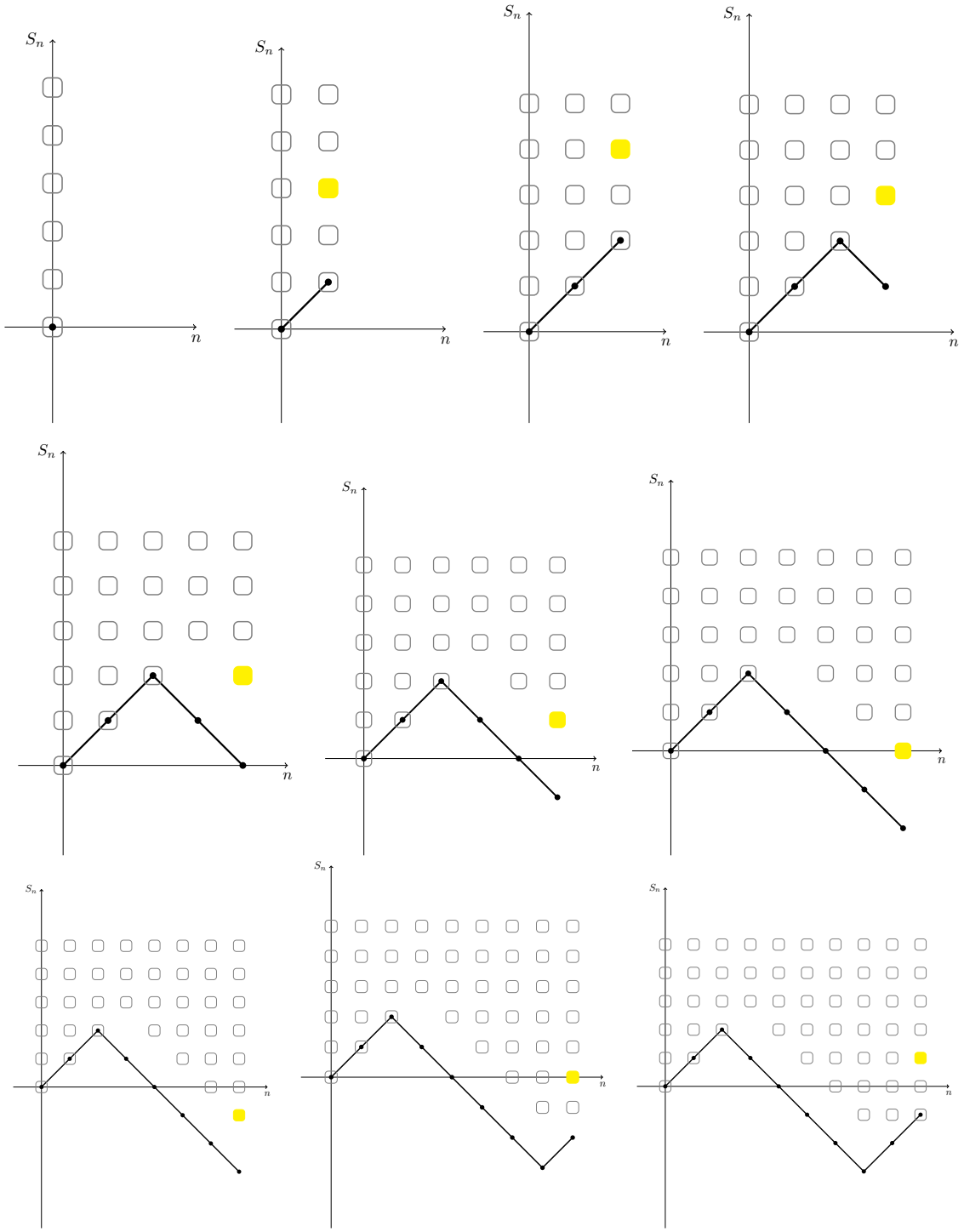

Figure 5.1: *Volume dynamics, yellow squares depict the position at distance* $\mu = 2$ *above the price* S_n *where the new order will be placed at the step* $n+1$.

6

Acronyms

AAPL	Apple company stock ticker symbol
ADF Test	Augmented Dickey-Fuller test
API	Application programming interface
CNN	Convolutional neural network
DLNN	Deep Learning Neural Network
GRU	Gated Recurrent Unit
HFT	High-Frequency Trading
iff	if and only if
KPI	key performance indicator
LOB	Limit Order Book
LOBSTER	Limit Order Book System Efficient Reconstructor
LSTM	Long Short Term Memory
MSFT	Microsoft company stock ticker symbol
NASDAQ	National Association of Securities Dealers Automated Quotations
PO	Price Oscillator
RNN	Recurrent neural network
RRR	risk-reward ratio
SUS	Stochastic Universal Sampling

Table 6.1: *Acronyms used in the thesis*

Bibliography

[1] Frédéric Abergel, Marouane Anane, Anirban Chakraborti, Aymen Jedidi, and Ioane Muni Toke. *Limit order books*. Cambridge University Press, 2016.

[2] Frédéric Abergel and Aymen Jedidi. A mathematical approach to order book modeling. *International Journal of Theoretical and Applied Finance*, 16(05):1350025, 2013.

[3] Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55 of *National Bureau of Standards Applied Mathematics Series*. For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C., 1964.

[4] Steven B Achelis. *Technical Analysis from A to Z*. McGraw Hill New York, 2001.

[5] Charu C Aggarwal. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

[6] Anna Aksamit, Tahir Choulli, and Monique Jeanblanc. Classification of random times and applications. 2016.

[7] Shahrokh Asadi. Evolutionary fuzzification of ripper for regression: Case study of stock prediction. *Neurocomputing*, 331:121–137, 2019.

[8] James E Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms*, volume 206, pages 14–21, 1987.

[9] Christian Bayer, Ulrich Horst, and Jinniao Qiu. A Functional Limit Theorem for Limit Order Books. *ArXiv e-prints*, May 2014.

[10] Hagan Demuth Beale, Howard B Demuth, and MT Hagan. Neural network design. *Pws, Boston*, 1996.

[11] Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[12] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[13] Jean Bertoin, Loïc Chaumont, and Jim Pitman. Path transformations of first passage bridges. *Electronic Communications in Probability*, 8:155–166, 2003.

[14] Patrick Billingsley. Convergence of probability measures john wiley & sons. *INC, New York*, 2(2.4), 1999.

[15] Zvi Bodie, Alex Kane, and Alan Marcus. Investments (10th global ed.). *Berkshire: McGraw-Hill Education*, 2014.

[16] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8, 2011.

[17] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[18] Dragos Bozdog, Ionut Florescu, Khaldoun Khashanah, and Jim Wang. Rare events analysis for high-frequency equity data. *Wilmott*, 2011(54):74–81, 2011.

[19] R.N. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill Kogakusha, Ltd., Tokyo, second edition, 1978.

[20] Janusz Brzeszczyński and Boulis Maher Ibrahim. A stock market trading system based on foreign and domestic information. *Expert Systems with Applications*, 118:381 – 399, 2019.

[21] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, pages 1–21, 2019.

[22] Álvaro Cartea, Sebastian Jaimungal, and José Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.

[23] S. Cascianelli, G. Costante, T. A. Ciarfuglia, P. Valigi, and M. L. Fravolini. Full-gru natural language video description for service robotics applications. *IEEE Robotics and Automation Letters*, 3(2):841–848, April 2018.

[24] Paola Cerchiello, Paolo Giudici, and Giancarlo Nicola. Twitter data models for bank risk contagion. *Neurocomputing*, 264:50–56, 2017.

[25] J Chang, Y Jung, K Yeon, J Jun, D Shin, and H Kim. Technical indicators and analysis methods. *Seoul: Jinritamgu Publishing*, 1996.

[26] Sotirios P. Chatzis, Vassilis Siakoulis, Anastasios Petropoulos, Evangelos Stavroulakis, and Nikos Vlachogiannakis. Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert Systems with Applications*, 112:353 – 371, 2018.

[27] Yingjun Chen and Yijie Hao. Integrating principle component analysis and weighted support vector machine for stock trading signals prediction. *Neurocomputing*, 321:381–402, 2018.

[28] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[29] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[30] Robert W Colby and Thomas A Meyers. *The encyclopedia of technical market indicators*. Dow Jones-Irwin Homewood, IL, 1988.

[31] Rama Cont and Adrien de Larrard. Price dynamics in a Markovian limit order market. *SIAM Journal on Financial Mathematics*, 4(1):1–25, 2013.

[32] Rama Cont, Arseniy Kukanov, and Sasha Stoikov. The price impact of order book events. *Journal of financial econometrics*, 12(1):47–88, 2014.

[33] Tony Cox. Algorithms to live by: The computer science of human decisions, brian christian and tom griffith. 2016. picador: New york, ny reviewed. *Risk Analysis*, 37(6):1201–1207, 2017.

[34] E. Csáki and S. G. Mohanty. Some joint distributions for conditional random walks. *The Canadian Journal of Statistics*, 14(1):19–28, 1986.

[35] Endre Csáki. Some joint distributions in Bernoulli excursions. *Journal of Applied Probability*, 31A:239–250, 1994. Studies in applied probability.

[36] Endre Csáki and Yueyun Hu. Lengths and heights of random walk excursions. In *Discrete random walks (Paris, 2003)*, Discrete Math. Theor. Comput. Sci. Proc., AC, pages 45–52. Assoc. Discrete Math. Theor. Comput. Sci., Nancy, 2003.

[37] Endre Csáki and Yueyun Hu. Invariance principles for ranked excursion lengths and heights. *Electronic Communications in Probability*, 9:14–21, 2004.

[38] Miklós Csörgő and Pál Révész. Long random walk excursions and local time. *Stochastic Processes and their Applications*, 41(2):181–190, 1992.

[39] Smruti Rekha Das, Debahuti Mishra, and Minakhi Rout. Stock market prediction using firefly algorithm with evolutionary framework optimized feature reduction for OSELM method. *Expert Systems with Applications: X*, 4:100016, 2019.

[40] L. Dudok de Wit. *Liquidity risks based on the limit order book*. Master thesis, Vienna University of Technology, 2013.

[41] Sylvain Delattre, Christian Y. Robert, and Mathieu Rosenbaum. Estimating the efficient price from the order flow: a Brownian Cox process approach. *Stochastic Processes and their Applications*, 123(7):2603–2619, 2013.

[42] Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. *Neural network design*. Martin Hagan, 2014.

[43] Rahul Dey and Fathi M Salemt. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[44] Min Ding, Hao Zhou, Hua Xie, Min Wu, Yosuke Nakanishi, and Ryuichi Yokoyama. A gated recurrent unit neural networks based wind speed error correction model for short-term wind power forecasting. *Neurocomputing*, 365:54 – 61, 2019.

[45] Matthew Dixon. Sequence classification of the limit order book using recurrent neural networks. *Journal of computational science*, 24:277–286, 2018.

[46] Matthew F Dixon, Nicholas G Polson, and Vadim O Sokolov. Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading. *arXiv preprint arXiv:1705.09851*, 2017.

[47] Rick Durrett. *Probability: theory and examples*. Cambridge University Press, Cambridge, fourth edition, 2010.

[48] Aniruddha Dutta, Saket Kumar, and Meheli Basu. A gated recurrent unit approach to bitcoin price prediction. *Journal of Risk and Financial Management*, 13(2):23, 2020.

[49] Ernst Eberlein, Kathrin Glau, and Antonis Papapantoleon. Analysis of fourier transform valuation formulas and applications. *Applied Mathematical Finance*, 17(3):211–240, 2010.

[50] Arthur Erdélyi, Wilhelm Magnus, Fritz Oberhettinger, and Francesco G Tricomi. Higher transcendental functions. *New York*, 1, 1955.

[51] William Feller. *An introduction to probability theory and its applications. Vol. I*. Third edition. John Wiley & Sons Inc., New York, 1968.

[52] William Feller. *An introduction to probability theory and its applications. Vol. II*. Second edition. John Wiley & Sons Inc., New York, 1971.

[53] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.

[54] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. cambridge University press, 2009.

[55] Peter Fleming and A Chipperfield. *Genetic algorithms in engineering systems*, volume 55. Iet, 1997.

[56] Hubalek Friedrich and Radojicic Dragana. On binomial order avalanches. *arXiv preprint arXiv:2007.07792*, 2020.

[57] Indranil Ghosh, Rabin K. Jana, and Manas K. Sanyal. Analysis of temporal pattern, causal interaction and predictive modeling of financial markets using nonlinear dynamics, econometric models and machine learning algorithms. *Applied Soft Computing*, 82:105553, 2019.

[58] Elli Gifford. *Investor's guide to technical analysis: predicting price action in the markets*. Financial Times, 1995.

[59] Mustafa Göçken, Mehmet Özçalıcı, Aslı Boru, and Ayşe Tuğba Dosdoğru. Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Systems with Applications*, 44:320–331, 2016.

[60] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[61] Martin D Gould, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. Limit order books. *Quantitative Finance*, 13(11):1709–1742, 2013.

[62] Aleksandra Grzesiek and Agnieszka Wyłomańska. Subordinated processes with infinite variance. In *Workshop on Cyclostationary Systems and Their Applications*, pages 111–135. Springer, 2017.

[63] L. Harris. *Trading and Exchanges: Market Microstructure for Practitioners*. Oxford University Press, Oxford, 2002.

[64] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[65] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[66] Ulrich Horst and Dörte Kreher. A weak law of large numbers for a limit order book model with fully state dependent order dynamics. *SIAM Journal on Financial Mathematics*, 8(1):314–343, 2017.

[67] Ulrich Horst and Michael Paulsen. A law of large numbers for limit order books. *Mathematics of Operations Research*, 42(4):1280–1312, 2017.

[68] Ehsan Hoseinzade and Saman Haratizadeh. Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273 – 285, 2019.

[69] Ruihong Huang and Tomas Polak. Lobster: Limit order book reconstruction system. *Available at SSRN 1977207*, 2011.

[70] Shian-Chang Huang, Chei-Chang Chiou, Jui-Te Chiang, and Cheng-Feng Wu. A novel intelligent option price forecasting and trading system by multiple kernel adaptive filters. *Journal of Computational and Applied Mathematics*, 369:112560, 2020.

[71] Thierry Jeulin. *Semi-martingales et grossissement d'une filtration*, volume 833. Springer, 2006.

[72] Ameet V Joshi. *Machine Learning and Artificial Intelligence*. Springer, 2020.

[73] Andrei Kirilenko, Albert S Kyle, Mehrdad Samadi, and Tugkan Tuzun. The flash crash: High-frequency trading in an electronic market. *The Journal of Finance*, 72(3):967–998, 2017.

[74] Lambert H Koopmans. *The spectral analysis of time series*. Elsevier, 1995.

[75] K Krishna and M Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999.

[76] Łukasz Kruk. Functional limit theorems for a simple auction. *Mathematics of Operations Research*, 28(4):716–751, 2003.

[77] Deepak Kumar, Suraj S Meghwani, and Manoj Thakur. Proximal support vector machine based hybrid prediction models for trend forecasting in financial markets. *Journal of Computational Science*, 17:1–13, 2016.

[78] Arthur le Calvez and Dave Cliff. Deep learning can replicate adaptive traders in a limit-order-book financial market. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1876–1883. IEEE, 2018.

[79] Tae Kyun Lee, Joon Hyung Cho, Deuk Sin Kwon, and So Young Sohn. Global stock market investment strategies based on financial network indicators using machine learning techniques. *Expert Systems with Applications*, 117:228 – 242, 2019.

[80] Yawen Li, Weifeng Jiang, Liu Yang, and Tian Wu. On neural networks and learning systems for business computing. *Neurocomputing*, 275:1150–1159, 2018.

[81] Andreas Lindell and Lars Holst. Distributions of the longest excursions in a tied down simple random walk and in a Brownian bridge. *Journal of Applied Probability*, 44(4):1056–1067, 2007.

[82] Jacob Loveless, Sasha Stoikov, and Rolf Waeber. Online algorithms in high-frequency trading. *Communications of the ACM*, 56(10):50–56, 2013.

[83] Mark M Meerschaert and Hans-Peter Scheffler. Limit theorems for continuous-time random walks with infinite mean waiting times. *Journal of applied probability*, 41(3):623–638, 2004.

[84] Mark M Meerschaert and Hans-Peter Scheffler. Triangular array limits for continuous time random walks. *Stochastic processes and their applications*, 118(9):1606–1633, 2008.

[85] Mark M Meerschaert and Peter Straka. Inverse stable subordinators. *Mathematical modelling of natural phenomena*, 8(2):1–16, 2013.

[86] Lukas Menkhoff. The use of technical analysis by fund managers: International evidence. *Journal of Banking & Finance*, 34(11):2573–2586, 2010.

[87] John J Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[88] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[89] R. Naranjo and M. Santos. A fuzzy decision system for money investment in stock markets based on fuzzy candlesticks pattern recognition. *Expert Systems with Applications*, 133:34 – 48, 2019.

[90] Frank WJ Olver, Daniel W Lozier, Ronald F Boisvert, and Charles W Clark. *NIST handbook of mathematical functions hardback and CD-ROM*. Cambridge university press, 2010.

[91] Shanoli Samui Pal and Samarjit Kar. Time series forecasting for stock market prediction through data discretization by fuzzistics and rule generation by rough set theory. *Mathematics and Computers in Simulation*, 162:18 – 30, 2019.

[92] Deepan Palguna and Ilya Pollak. Mid-price prediction in a limit order book. *IEEE Journal of Selected Topics in Signal Processing*, 10(6):1083–1092, 2016.

[93] Robert Pardo. *The evaluation and optimization of trading strategies*, volume 314. John Wiley & Sons, 2011.

[94] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert systems with applications*, 42(1):259–268, 2015.

[95] Mihael Perman and Jon A. Wellner. An excursion approach to maxima of the Brownian bridge. *Stochastic Processes and their Applications*, 124(9):3106–3120, 2014.

[96] Songqiao Qi, Kaijun Jin, Baisong Li, and Yufeng Qian. The exploration of internet finance by using neural network. *Journal of Computational and Applied Mathematics*, 369:112630, 2020.

[97] D. Radojicic, N. Radojicic, and S. Kredatus. A multicriteria optimization approach for the stock market feature selection. *in preparation*, 2020.

[98] Dragana Radojičić and Simeon Kredatus. An approach for processing data from NASDAQ stock exchange database. In *Proceedings of the 10th International Conference on Information Society and Technology (ICIST 2020)*. Proceedings Vol.2, pp.256-259, 2020, 2020.

[99] Dragana Radojičić, Simeon Kredatus, and Thorsten Rheinländer. An approach to reconstruction of data set via supervised and unsupervised learning. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 000053–000058. IEEE, 2018.

[100] Dragana Radojičić and Simeon Kredatus. The impact of stock market price fourier transform analysis on the gated recurrent unit classifier model. *Expert Systems with Applications*, 159:113565, 2020.

[101] Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*. Springer-Verlag, Berlin, third edition, 1999. (in particular Ch. XII: Excursion theory).

[102] F. Riccardi. *Stochastic Models for the Limit Order Book*. Mphil thesis, London School of Economics, 2013.

[103] Samuel Rönnqvist and Peter Sarlin. Bank distress in the news: Describing events through deep learning. *Neurocomputing*, 264:57–70, 2017.

[104] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[105] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

[106] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[107] Yauheniya Shynkevich, T Martin McGinnity, Sonya A Coleman, Ammar Belatreche, and Yuhua Li. Forecasting price movements using technical indicators: Investigating the impact of varying input window length. *Neurocomputing*, 264:71–88, 2017.

[108] Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, 2019.

[109] Laurel Smith and Persi Diaconis. Honest Bernoulli excursions. *Journal of Applied Probability*, 25(3):464–477, 1988.

[110] Sabine Sporer. *Verteilungen im Rahmen des Limit-Order-Buches*. Master thesis, Vienna University of Technology, 2014.

[111] M. A. Stapleton and K. Christensen. One-dimensional directed sandpile models and the area under a Brownian curve. *Journal of Physics. A. Mathematical and General*, 39(29):9107–9126, 2006.

[112] Lajos Takács. Brownian local times. *Journal of Applied Mathematics and Stochastic Analysis*, 8(3):209–232, 1995.

[113] Lajos Takács. The distribution of the sojourn time for the Brownian excursion. *Methodology and Computing in Applied Probability*, 1(1):7–28, 1999.

[114] Dongge Tang, Wenge Rong, Shuang Qin, Jianxin Yang, and Zhang Xiong. A n-gated recurrent unit with review for answer selection. *Neurocomputing*, 371:158–165, 2020.

[115] Igor V. Tetko, David J. Livingstone, and Alexander I. Luik. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences*, 35(5):826–833, 1995.

[116] Joy A Thomas and TM Cover. Elements of information theory. *John Wiley & Sons, Inc., New York. Toni, T., Welch, D., Strelkowa, N., Ipsen, A., and Stumpf, MPH (2009),"Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems," Journal of the Royal Society Interface*, 6:187–202, 1991.

[117] Marcela Valenzuela, Ilknur Zer, Piotr Fryzlewicz, and Thorsten Rheinländer. Relative liquidity and future volatility. Technical Report 45, Federal Reserve Board, Washington, D.C., 2014.

[118] Vince Vella and Wing Lon Ng. Enhancing risk-adjusted performance of stock market intraday trading with neuro-fuzzy systems. *Neurocomputing*, 141:170–187, 2014.

[119] Jorge R Vergara and Pablo A Estévez. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24(1):175–186, 2014.

[120] Wim Vervaat. A relation between Brownian bridge and Brownian excursion. *The Annals of Probability*, 7(1):143–149, 1979.

[121] Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision rnns for language recognition. *arXiv preprint arXiv:1805.04908*, 2018.

[122] Ward Whitt. *Stochastic-process limits: an introduction to stochastic-process limits and their application to queues*. Springer Science & Business Media, 2002.

[123] Qifa Xu, Liukai Wang, Cuixia Jiang, and Xin Zhang. A novel umidas-svqr model with mixed frequency investor sentiment for predicting stock market volatility. *Expert Systems with Applications*, 132:12 – 27, 2019.

[124] D. Zhang, L. Tian, M. Hong, F. Han, Y. Ren, and Y. Chen. Combining convolution neural network and bidirectional gated recurrent unit for sentence semantic classification. *IEEE Access*, 6:73750–73759, 2018.

[125] Xi Zhang, Jiawei Shi, Di Wang, and Binxing Fang. Exploiting investors social network for stock prediction in china's market. *Journal of computational science*, 28:294–303, 2018.

[126] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang. Machine health monitoring using local feature-based gated recurrent unit networks. *IEEE Transactions on Industrial Electronics*, 65(2):1539–1548, 2018.

[127] Ban Zheng, Eric Moulines, and Frédéric Abergel. Price jump prediction in limit order book. *arXiv preprint arXiv:1204.1381*, 2012.

[128] Zeqi Zheng, Yuandong Gao, Likang Yin, and Monika K. Rabarison. Modeling and analysis of a stock-based collaborative filtering algorithm for the chinese stock market. *Expert Systems with Applications*, page 113006, 2019.

[129] Feng Zhou, Qun Zhang, Didier Sornette, and Liu Jiang. Cascading logistic regression onto gradient boosted decision trees for forecasting and trading stock indices. *Applied Soft Computing*, 84:105747, 2019.

[130] Oleksandr Zhylyevskyy. A fast fourier transform technique for pricing american options under stochastic volatility. *Review of Derivatives Research*, 13(1):1–24, 2010.