

https://dspace.org/

An online adaptive learning algorithm for optimal trade execution in high-frequency markets

Hendricks, Dieter

2016

http://hdl.handle.net/10539/21710

Hendricks, Dieter (2016) An online adaptive learning algorithm for optimal trade execution in high-frequency markets, University of Witwatersrand, Johannesburg, <http://wiredspace.wits.ac.za/handle/10539/21710>

Downloaded from a DSpace repository. https://dspace.org/



An online adaptive learning algorithm for optimal trade execution in high-frequency markets

Author:
Dieter Hendricks

Supervisors:
Prof. Diane Wilcox
Dr. Tim Gebbie

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

in the

Faculty of Science
School of Computer Science and Applied Mathematics
University of the Witwatersrand

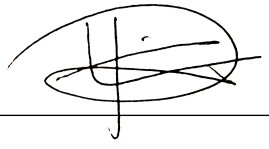
October 2016

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are those of the author and are not necessarily attributed to the NRF.

Declaration of Authorship

I, Dieter Hendricks, declare that this thesis titled, 'An online adaptive learning algorithm for optimal trade execution in high-frequency markets' and the work presented in it are my own. I confirm that:

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

Signed: 

Date: 04/10/2016

Far better an approximate answer to the right question, which is often vague, than the exact answer to the wrong question, which can always be made precise.

John Tukey, 1962.

UNIVERSITY OF THE WITWATERSRAND

Abstract

Faculty of Science
School of Computer Science and Applied Mathematics
Doctor of Philosophy

An online adaptive learning algorithm for optimal trade execution in high-frequency markets

by Dieter Hendricks

Automated algorithmic trade execution is a central problem in modern financial markets, however finding and navigating optimal trajectories in this system is a non-trivial task. Many authors have developed exact analytical solutions by making simplifying assumptions regarding governing dynamics, however for practical feasibility and robustness, a more dynamic approach is needed to capture the spatial and temporal system complexity and adapt as intraday regimes change.

This thesis aims to consolidate four key ideas: 1) the financial market as a complex adaptive system, where purposeful agents with varying system visibility collectively and simultaneously create and perceive their environment as they interact with it; 2) spin glass models as a tractable formalism to model phenomena in this complex system; 3) the multivariate Hawkes process as a candidate governing process for limit order book events; and 4) reinforcement learning as a framework for online, adaptive learning. Combined with the data and computational challenges of developing an efficient, machine-scale trading algorithm, we present a feasible scheme which systematically encodes these ideas.

We first determine the efficacy of the proposed learning framework, under the conjecture of approximate Markovian dynamics in the equity market. We find that a simple lookup table *Q-learning* algorithm, with discrete state attributes and discrete actions, is able to improve post-trade implementation shortfall by adapting a typical static arrival-price volume trajectory with respect to prevailing market microstructure features streaming from the limit order book.

To enumerate a scale-specific state space whilst avoiding the *curse of dimensionality*, we propose a novel approach to detect the intraday temporal financial market state at each decision point in the *Q-learning* algorithm, inspired by the complex adaptive system paradigm. A physical analogy to the ferromagnetic Potts model at thermal equilibrium is used to develop a high-speed maximum likelihood clustering algorithm, appropriate for measuring critical or near-critical temporal states in the financial system. State

features are studied to extract time-scale-specific *state signature vectors*, which serve as low-dimensional state descriptors and enable online state detection.

To assess the impact of agent interactions on the system, a multivariate Hawkes process is used to measure the resiliency of the limit order book with respect to liquidity-demand events of varying size. By studying the branching ratios associated with key quote replenishment intensities following trades, we ensure that the limit order book is expected to be resilient with respect to the maximum permissible trade executed by the agent.

Finally we present a feasible scheme for unsupervised state discovery, state detection and online learning for high-frequency quantitative trading agents faced with a multi-featured, asynchronous market data feed. We provide a technique for enumerating the state space at the scale at which the agent interacts with the system, incorporating the effects of a live trading agent on limit order book dynamics into the market data feed, and hence the perceived state evolution.

Acknowledgements

I would like to firstly thank Prof. Diane Wilcox and Dr. Tim Gebbie, whose breadth of knowledge, keen insight, appreciation for rigour and unrelenting support provided an indispensable platform for success during my PhD journey. I also thank Dr. Nicholas Westray and Dr. Anja Richter, who entertained many discussions during the conceptual foundations of this work and were more than generous and accommodating during my visits to London and New York.

I thank the Fields Institute for Research in the Mathematical Sciences for hosting me during my participation at the thematic program on Statistical Inference, Learning and Models in Big Data. I learnt an immense amount from the workshops and had time to focus on my core proposition in this thesis and contextualise the contribution. I am grateful to Prof. Nancy Reid for inviting me as a funded visitor for these crucial 3 months.

I thank the Instituto Nacional de Matemática Pura e Aplicada (IMPA) for hosting me during my participation at the thematic program on Stochastic Variational Analysis, where I had the time and space to put the final touches on this thesis. Thank you to Dr. Claudia Sagastizabal and Dr. Welington de Oliveira for funding my time at IMPA.

I thank Ms. Waheeda Bala and the National Research Foundation of South Africa for generously awarding me a doctoral scholarship (Grant ID: 89250) for the duration of my studies, and providing travel support for related conference visits.

I am extremely grateful to my Wits School of CSAM colleagues for their administrative (and emotional) support. The school provided a fantastic environment for collaborative discussion, work-related or otherwise. In no particular order, I thank: Prof. Ebrahim Momoniat, Prof. Charis Harley, Prof. Turgay Celik, Dr. Byron Jacobs, Dr. Michael Mitchley, Dr. Rhameez Herbst, Mr. Michael Harvey, Mr. Kedy Mazibuko, Mr. Roger Martins, Ms. Precious Shabalala, Ms. Keba Mosiane and Ms. Dorina Bowes.

Of course, I thank my family, who continually provide a source of inspiration and support, despite my varied academic pursuits and random musings.

Lastly, I thank the music community of Gauteng, who embraced me when I moved to Johannesburg and allowed me to participate in the vibrant classical music scene - an essential balance-maintaining component whilst undertaking this research, but more importantly affirming music's place in my professional pursuits going forward. Thank you to the Johannesburg Philharmonic Orchestra, Johannesburg Festival Orchestra, Agathe String Sextet, DiSetinel String Quartet and numerous other fantastic chamber groups.

Preface

This thesis combines the contributions of the following papers written over the duration of my PhD registration, which will be extensively referenced and expanded upon, with certain sections taken verbatim from the associated paper text. With respect to co-authors, Prof. Diane Wilcox and Dr. Tim Gebbie were my PhD supervisors, and Mr. Michael Harvey and Mr. Roger Martins were students which I supervised, providing related topics for their major honours projects which we co-investigated. The project topics were chosen based on specific investigations which formed part of my PhD work.

- D. Hendricks, D. Wilcox. *A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution.* Proceedings from IEEE Conference on Computational Intelligence for Financial Economics and Engineering, 2014. [124]
  Available online: http://dx.doi.org/10.1109/CIFEr.2014.6924109

- D. Hendricks, T. Gebbie, D. Wilcox. *High-speed detection of emergent market clustering via an unsupervised parallel genetic algorithm.* South African Journal of Science, vol. 112, no. 1/2, 2016. [125]
  Available online: http://dx.doi.org/10.17159/sajs.2016/20140340

- D. Hendricks, T. Gebbie, D. Wilcox. *Detecting intraday financial market states using temporal clustering.* Quantitative Finance, 2016. [126]
  Available online: http://dx.doi.org/10.1080/14697688.2016.1171378

- D. Hendricks, M. Harvey. *Reconciling order book resiliency and price impact.* Working paper, 2016. [123]

- R. Martins, D. Hendricks. *The statistical significance of multivariate Hawkes processes fitted to limit order book data.* Working paper (submitted to Journal of Applied Probability, under review), 2016. [174]

- D. Hendricks. *An online learning algorithm with scale-specific state space enumeration for optimal trade execution in high-frequency markets.* Working paper, 2016. [121]

- D. Hendricks. *Using real-time cluster configurations of streaming asynchronous features as online state descriptors in financial markets.* Working paper (submitted to Pattern Recognition Letters, under review), 2016. [122]

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **v**

Preface **vi**

Contents **vii**

List of Figures **xi**

List of Tables **xviii**

Abbreviations **xx**

1 Introduction and overview **1**
1.1 The financial market as a complex adaptive system 1
1.2 Spin glass models for modelling complex system behaviour 4
1.3 The Hawkes process as the governing process for microstructure events . . 6
1.4 Reinforcement learning as a framework for online, adaptive trajectories through the complex system 7
1.5 Automated algorithmic trading in modern financial markets 9
1.6 Overview 10

2 Market microstructure and the trade execution problem **12**
2.1 Overview 12
2.2 Market microstructure 12
2.3 The limit order book and trading mechanism 14
2.4 Price impact 17
2.5 Order book resiliency 18
2.6 Optimal trade execution 20
2.7 Some remarks 22

3 Model-free reinforcement learning 23
3.1 Overview . . . 23
3.2 Markov Decision Processes . . . 24
3.3 Dynamic Programming . . . 26
3.3.1 Policy iteration . . . 26
3.3.2 Value iteration . . . 27
3.4 The *Q-learning* algorithm . . . 28
3.4.1 Proof of convergence for infinite-horizon *Q-learning* . . . 29
3.4.2 On convergence for finite-horizon *Q-learning* . . . 33
3.5 Batch learning vs online learning . . . 34
3.6 Exploration vs exploitation trade-off . . . 35
3.7 Curse of dimensionality . . . 35
3.8 The nature of learning in a complex system . . . 36
3.9 Some remarks . . . 37

4 Data description and Exploratory Data Analysis 38
4.1 Overview . . . 38
4.2 Data . . . 38
4.2.1 Raw data . . . 38
4.2.2 MongoDB noSQL database . . . 40
4.2.2.1 Query indexes . . . 41
4.2.2.2 Aggregation and Map-Reduce . . . 42
4.2.2.3 MATLAB API . . . 42
4.3 Exploratory Data Analysis . . . 43
4.3.1 Visualisation of limit order book features . . . 44
4.4 Some remarks . . . 52

5 A simple model-free reinforcement learning model for trade execution 53
5.1 Overview . . . 53
5.2 Adapting a static liquidation trajectory using reinforcement learning . . . 54
5.2.1 The Almgren-Chriss model for optimal liquidation . . . 55
5.2.2 State space . . . 58
5.2.3 Action set . . . 60
5.2.4 Reward function . . . 61
5.2.5 Algorithm . . . 62
5.3 Data and results . . . 64
5.3.1 Data used . . . 64
5.3.2 Stocks, parameters and assumptions . . . 64
5.3.3 Results . . . 65
5.4 Some remarks . . . 68

6 Detecting intraday states from streaming market microstructure features 70
6.1 Overview . . . 70
6.2 From unsupervised clustering to temporal states . . . 70
6.3 Super-paramagnetic clustering for state discovery and detection . . . 72
6.3.1 Potts spin models as analogue for financial system . . . 72

6.4 A maximum likelihood approach . . . 73
6.5 Considering time periods as objects for market state determination . . . 76
6.6 State Signature Vectors for online state detection . . . 76
6.7 Scale-invariant characteristics of states . . . 78
6.8 A high-speed Parallel Genetic Algorithm implementation . . . 79
6.8.1 GA principle and genetic operators . . . 80
6.8.2 Master-slave parallelisation . . . 81
6.8.3 Computational Platform and Implementation . . . 82
6.8.3.1 Specific computational environment . . . 83
6.8.3.2 Implementation . . . 83
6.8.3.3 Representation . . . 84
6.8.3.4 Fitness function . . . 84
6.8.3.5 Master-slave PGA implementation . . . 84
6.8.3.6 Key implementation challenges . . . 87
6.9 Results . . . 88
6.9.1 Data description . . . 88
6.9.2 Workflow . . . 89
6.9.3 Visualisation . . . 90
6.9.4 Results discussion . . . 91
6.10 Identifying high-frequency persistent states using *event-time clustering* . . . 105
6.11 Some remarks . . . 108

7 Using order book resiliency to control agent actions 110
7.1 Overview . . . 110
7.2 Modelling order book resiliency . . . 111
7.2.1 Multivariate Hawkes process for limit order book events . . . 111
7.2.2 Enumerating empirical event point processes using tick data . . . 113
7.2.2.1 Volume-conditional liquidity demand point processes . . . 116
7.2.3 Candidate kernels for encoding temporal dependence . . . 118
7.2.4 Deriving maximum likelihood estimator with sum-of-exponentials kernel . . . 120
7.2.5 Calibration of model parameters . . . 122
7.2.6 On the choice of M (number of exponentials) . . . 124
7.2.7 Motivating use of time-dependent baseline intensity . . . 125
7.2.8 An efficient non-parametric calibration scheme . . . 127
7.3 Effect of volume-conditional trade events on quote replenishment intensity 128
7.4 Some remarks . . . 131

8 Using detected states and resilient actions to enhance the trade execution algorithm 132
8.1 Overview . . . 132
8.2 Recall the basic reinforcement learning model . . . 133
8.3 Using temporal state as market attribute . . . 134
8.4 Bounding actions using resiliency . . . 135
8.5 On the learning rate . . . 135
8.6 Algorithm . . . 136
8.7 Data and Results . . . 137

8.7.1 Data . . . 137
8.7.2 Results . . . 137
8.8 Some remarks . . . 139

9 Towards unsupervised, online state discovery, detection and learning in high-frequency financial markets 140
9.1 Overview . . . 140
9.2 Representation learning for tractable inference in high-dimensional state spaces . . . 141
9.3 Cluster configurations as temporal state descriptors . . . 142
9.4 Correlation estimation from streaming asynchronous data . . . 145
9.5 High-speed feature clustering . . . 146
9.6 Cluster configuration similarity and state discrimination . . . 147
9.7 Reinforcement learning with online state discovery . . . 149
9.8 Problem description and Algorithm . . . 150
9.8.1 Wealth maximisation: *Long-only* . . . 150
9.8.2 Algorithm . . . 151
9.9 Data and Results . . . 152
9.9.1 Data . . . 152
9.9.2 Results . . . 153
9.10 Some remarks . . . 159

10 Conclusion 160

A Derivation of the maximum likelihood function for explanatory power of cluster configuration 165
A.1 The Noh-Giada-Marsili coupling parameters . . . 165
A.2 The Noh-Giada-Marsili likelihood function . . . 167

Bibliography 170

List of Figures

2.1 Illustrating the effect of liquidity demand and subsequent replenishment. A buy *market order* arrives and removes commensurate quotes from the ask side of the limit order book (2), creating a transient deficit (3). Bid and ask limit orders are then submitted by other participants (4,5) and liquidity is restored (6). This figure is reproduced from the text, *Limit Order Books* by Abergel et al. [4]. 16

2.2 Some stylised aspects of a limit order book. Green lines indicate *ask quotes* and blue lines indicate *bid quotes*, with the vertical position showing the price level and line length showing quote quantity. This figure illustrates market depth, quote imbalance and temporary price impact. 17

2.3 Trade execution with arrival price benchmark 21

4.1 Implemented MongoDB schema for TRTH data. A *TickData* database was created to store the data in two *collections*: *JSETransactions*, which stores trade and level-1 quotes, and *JSEMarketDepth*, which stores 10 levels of market depth quotes. 40

4.2 Some compound indexes created for efficient query execution. 41

4.3 Aggregation pipeline versus map-reduce for a simple computation, extracting level-1 quotes for a stock at 5-minute intervals. 43

4.4 This figure aims to demonstrate the asynchronous nature of the trade price time series at the tick level. Here, we plot raw trades of all TOP40 stocks over a 5 minute period, from 09:00 02 October 2012 to 09:05 02 October 2012. Each dot represents a trade at the exact time it took place, with trade price on the Y-axis and time on the X-axis, where the size of the dot is proportional to the volume of the trade. Dots are coloured by stock. It is clear stock trades occur asynchronously. 45

4.5 This figure aims to investigate common temporal patterns amongst stock trades across the trading day. Here, we plot raw trades of all TOP40 stocks over a 7.5 hour period, from 09:00 02 October 2012 to 16:30 02 October 2012. Each dot represents a trade at the exact time it took place, with trade price on the Y-axis and time on the X-axis, where the size of the dot is proportional to the volume of the trade. Dots are coloured by stock. 46

4.6 This figure aims to investigate the nature of spreads by plotting the evolution of level-1 quotes. 47

4.7 This figure aims to investigate the nature of spreads by plotting the evolution of level-1 quotes. 47

4.8 This figure aims to investigate the nature of level-1 quote updates for two fundamentally similar stocks which are typical candidates for pairs trading, Mondi Limited (MND) and Mondi Plc (MNP). 48

4.9 This figure aims to investigate the nature of the raw events of interest in the limit order book, including market depth. Ask quotes are indicated in blue and bid quotes in red, with darker colours indicating closeness to the top-of-the-book. Yellow dots indicate trade events. The size of each dot is proportional to the volume of the trade or quote. 49
4.10 This figure aims to investigate the nature of the raw events of interest in the limit order book, including market depth. Ask quotes are indicated in blue and bid quotes in red, with darker colours indicating closeness to the top-of-the-book. Yellow dots indicate trade events. The size of each dot is proportional to the volume of the trade or quote. 50
4.11 This figure aims to investigate the nature of the raw events of interest in the limit order book, including market depth. Ask quotes are indicated in blue and bid quotes in red, with darker colours indicating closeness to the top-of-the-book. Yellow dots indicate trade events. The size of each dot is proportional to the volume of the trade or quote. 51

5.1 Difference between median implementation shortfall generated using RL and AC models, with given parameters ($I,B,W = 5$). Training H-dependent. 67
5.2 % correct actions implied by Q-matrix after each training set tuple. Training H-dependent. 68

6.1 Illustration of online state assignment based on identified state signature vectors. 77
6.2 Mapping of individuals onto the CUDA thread hierarchy 86
6.3 Flowchart illustrating workflow to determine the temporal cluster configuration from a time period correlation matrix, identify persistent states, estimate temporal cluster configuration using feature vectors and determine state transition probabilities. Processes are coloured by platform: MongoDB = Yellow, MATLAB = Green, CUDA-C = Orange, Gephi = Purple. 90
6.4 JSE TOP40 60-minute temporal clusters for the period 01-Nov-2012 to 30-Nov-2012, representing 184 distinct periods. Each node represents a 60-minute period during a trading day, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership. 95
6.5 JSE TOP40 60-minute cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses. 95
6.6 JSE TOP40 30-minute temporal clusters for the period 01-Nov-2012 to 30-Nov-2012, representing 368 distinct periods. Each node represents a 30-minute period during a trading day, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership. 96

6.7 JSE TOP40 30-minute cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses. 96
6.8 JSE TOP40 15-minute temporal clusters for the period 01-Nov-2012 to 30-Nov-2012, representing 736 distinct periods. Each node represents a 15-minute period during a trading day, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership. 97
6.9 JSE TOP40 15-minute cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses. 97
6.10 JSE TOP40 5-minute temporal clusters for the period 01-Nov-2012 to 30-Nov-2012, representing 2208 distinct periods. Each node represents a 5-minute period during a trading day, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership. 98
6.11 JSE TOP40 5-minute cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses. 98
6.12 Testing conjecture of power law fit for varying time scale cluster sizes, applying the Clauset, Shalizi and Newman algorithm [58]. α indicates the scaling parameter of the proposed fit, p_{value} indicates the p-value from a Kolmogorov-Smirnov test for the goodness-of-fit of the proposed model to the data, x_{min} indicates the lower-bound for the power law fit and $\mathcal{L}$ is the log-likelihood of the data ($x \geq x_{min}$) under the power law fit. 99
6.13 Estimated 60-minute clusters using identified state signature vectors. The Euclidean distance is calculated between each temporal period's feature vector and the state signature vectors. Cluster index assignment is based on the state signature vector which yields the minimum distance. 100
6.14 Estimated 30-minute clusters using identified state signature vectors. The Euclidean distance is calculated between each temporal period's feature vector and the state signature vectors. Cluster index assignment is based on the state signature vector which yields the minimum distance. 101
6.15 Estimated 15-minute clusters using identified state signature vectors. The Euclidean distance is calculated between each temporal period's feature vector and the state signature vectors. Cluster index assignment is based on the state signature vector which yields the minimum distance. 102

6.16 Estimated 5-minute clusters using identified state signature vectors. The Euclidean distance is calculated between each temporal period's feature vector and the state signature vectors. Cluster index assignment is based on the state signature vector which yields the minimum distance. 103
6.17 Measuring the stability of the online state assignment algorithm out-of-sample. Given that the state assignment of an online FV is based on the minimum Euclidean distance to predetermined SSVs, we compute the *best match* distance for each of the FVs in a sample and use a boxplot to visualise the empirical distribution. In this figure, we compare the *ex-ante* (01-Nov-2012 to 30-Nov-2012, same period used for SSV estimation) and *ex-post* (03-Dec-2012 to 07-Dec-2012, one week after SSV estimation window) periods. 104
6.18 JSE TOP40 event-time clusters for the period 01-Nov-2012 to 30-Nov-2012. Each node represents a **traded volume of 100 000 SHF shares**, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership. 106
6.19 JSE TOP40 **100 000 SHF volume** cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses. 106
6.20 JSE TOP40 event-time clusters for the period 01-Nov-2012 to 30-Nov-2012. Each node represents a **traded volume of 100 000 AGL shares**, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership. 107
6.21 JSE TOP40 **100 000 AGL volume** cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses. 107
6.22 Testing conjecture of power law fit for varying time scale cluster sizes, applying the Clauset, Shalizi and Newman algorithm [58]. α indicates the scaling parameter of the proposed fit, p_{value} indicates the p-value from a Kolmogorov-Smirnov test for the goodness-of-fit of the proposed model to the data, x_{min} indicates the lower-bound for the power law fit and $\mathcal{L}$ is the log-likelihood of the data ($x \geq x_{min}$) under the power law fit. 108

7.1 Empirical event intensities for SBK of each of the 4 key event types over a **morning period**, demonstrating event clustering and mutual-excitation. The dots show the arrival times of the events and the lines show the 5-minute event intensities. 115
7.2 Empirical event intensities for SBK of each of the 4 key event types over a **midday period**, demonstrating event clustering and mutual-excitation. The dots show the arrival times of the events and the lines show the 5-minute event intensities. 116

7.3 Empirical event intensities for SBK of each of the 4 key event types over an **afternoon period**, demonstrating event clustering and mutual-excitation. The dots show the arrival times of the events and the lines show the 5-minute event intensities. 116
7.4 Empirical **volume-conditional Type-1 event** intensities for SBK for 4 candidate volume bins over a typical trading day. The dots show the arrival times of the events and the lines show the 5-minute event intensities. 118
7.5 Empirical **volume-conditional Type-2 event** intensities for SBK for 4 candidate volume bins over a typical trading day. The dots show the arrival times of the events and the lines show the 5-minute event intensities. 118
7.6 Computation time (in minutes): vectorisation vs for-loop. Shows average computation time as a function of the number of each event type. 123
7.7 Computation time (in minutes): vectorisation vs for-loop. Shows average computation time as a function of the number of exponentials (M). 123
7.8 Average hourly baseline intensity for all events. Blue line indicates average for a given hour, with the error bars reflecting the variation over the days in the sample. 126
7.9 Baseline intensity by kernel. The trading day is divided into 3 periods (*morning*, *noon*, *afternoon*), with piecewise linear intensity. Coloured lines correspond to mean exogenous intensities for the given periods, calibrated for each kernel, with the error bars reflecting daily variation. . . . 127
7.10 Boxplot of distribution of calibrated branching ratios of volume-conditional Hawkes process, quantifying the **effect of aggressive buy trade** (Type 1) events of varying size on **aggressive ask quote** (Type 4) replenishment intensity. X-axis labels indicate upper bounds of trade volume bins, where volumes have been normalised by their mean. We thus see the resulting branching ratios for aggressive ask quote intensity, given buy trades with size as increasing multiples of the mean trade size. 130
7.11 Boxplot of distribution of calibrated branching ratios of volume-conditional Hawkes process, quantifying the **effect of aggressive sell trade** (Type 2) events of varying size on **aggressive bid quote** (Type 3) replenishment intensity. X-axis labels indicate upper bounds of trade volume bins, where volumes have been normalised by their mean. We thus see the resulting branching ratios for aggressive bid quote intensity, given sell trades with size as increasing multiples of the mean trade size. 130

8.1 Difference (RL - AC) in median *implementation shortfall* for various learning rates. The best *spread, volume* model is highlighted by the thick green line and the best *SSV* model is highlighted by the thick red line. 138

9.1 Illustrating how identified feature configurations may have an analogous interpretation, in terms of human-specified pre-processed features. 144

9.2 *Status at 09:35.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair. 154

9.3 *Status at 09:45.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair. 155

9.4 *Status at 10:30.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair. 156

9.5 *Status at 12:45.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair. 157

9.6 *Status at 13:30.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair. 158

List of Tables

4.1 Thomson Reuters Tick History (TRTH) raw data fields. The associated data was downloaded as flat text files from the TRTH web interface. . . . 39
4.2 Hardware specifications for MongoDB data server. 41

5.1 Average % improvement in median *implementation shortfall* for various parameter values, using AC and RL models. Training H-dependent. . . . 66
5.2 Standard deviation(%) of implementation shortfall when using AC vs RL models. 66

6.1 Development, testing and benchmarking environments 83
6.2 Restrictions on number of stocks and population size. For the Tesla card, *Max number of stocks* $= (14)*(1024/32)*8 = 3584$ and *Max population size* $= (65535/(3584/32)) * 32 = 18720$. 86
6.3 Illustration of data returns matrix as an input for estimation of 15-minute period correlations 89
6.4 Parameter values and computation times for Parallel Genetic Algorithm ∗ Average from 20 independent runs; N refers to the GTX765m Notebook GPU and D refers to the GTX Titan X Desktop GPU. 94
6.5 Empirical 1-step transition probability matrix for 60-minute states, based on identified temporal cluster configuration. State transitions with a probability > 0 are highlighted in green. 100
6.6 Empirical 1-step transition probability matrix for 30-minute states, based on identified temporal cluster configuration. State transitions with a probability > 0 are highlighted in green. 101
6.7 Empirical 1-step transition probability matrix for 15-minute states, based on identified temporal cluster configuration. State transitions with a probability > 0 are highlighted in green. 102
6.8 Empirical 1-step transition probability matrix for 5-minute states, based on identified temporal cluster configuration. State transitions with a probability > 0 are highlighted in green. 103

7.1 Daily goodness-of-fit test statistics by kernel, **GRT 01 September 2013 to 27 September 2013** 125
7.2 Daily goodness-of-fit test statistics by kernel, **GRT 30 September 2013 to 31 October 2013** 125

8.1 State attributes for simple RL trading agent. 133
8.2 Difference (RL - AC) in median *implementation shortfall* for various learning rates. The best *spread, volume* model is highlighted in green and the best *SSV* model is highlighted in red. 138

9.1 Demonstration of *best match* metric for calculating distance between two overlapping cluster configurations 148
9.2 Parameters used for testing long-only wealth maximisation algorithm. . . 151
9.3 Summarised results from algorithm testing. The *LO Wealth Maximiser* agent is compared to a *Random* agent, where actions are chosen randomly at each trading opportunity. The algorithm begins at 09:05 and ends at 16:30 each trading day. These results summarise the distribution of end-of-day (16:30) PnL recorded for each day in the investigation period (01 Oct 2012 to 30 Nov 2012). 153

Abbreviations

AC	**A**lmgren-**C**hriss
ALSI	**AL**l **S**hare **I**ndex
API	**A**pplication **P**rogramming **I**nterface
bps	**b**asis **p**oint**s**
CC	**C**ompute **C**apability
CPU	**C**entral **P**rocessing **U**nit
CNN	**C**onvolutional **N**eural **N**etwork
CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
DB	**D**ata **B**ase
DBMS	**D**ata **B**ase **M**anagement **S**ystem
ED	**E**xcess **D**ispersion
EWA	**E**xperience **W**eighted **A**ttraction
FV	**F**eature **V**ector
GA	**G**enetic **A**lgorithm
GB	**G**iga **B**yte
GM	**G**iada-**M**arsili
GPU	**G**raphics **P**rocessing **U**nit
HDD	**H**ar**D** **D**rive
HY	**H**ayashi-**Y**oshida
IRL	**I**nverse **R**einforcement **L**earning
IS	**I**mplementation **S**hortfall
JDBC	**J**ava **D**ata **B**ase **C**onnectivity
JSE	**J**ohannesburg **S**tock **E**xchange
JSON	**J**ava**S**cript **O**bject **N**otation
KPSS	**K**wiatkowski-**P**hillips-**S**chmidt-**S**hin

KS	**K**olmogorov **S**mirnov
LBQ	**L**jung **B**ox **Q**-test
LOB	**L**imit **O**rder **B**ook
LQ	**L**ower **Q**uartile
MDP	**M**arkov **D**ecision **P**rocess
MIT	**M**illennium **I**nformation **T**echnologies
MLE	**M**aximum **L**ikelihood **E**stimator
MM	**M**alliavin-**M**ancino
MPI	**M**essage **P**arsing **I**nterface
OS	**O**perating **S**ystem
PGA	**P**arallel **G**enetic **A**lgorithm
PnL	**P**rofit a**n**d **L**oss
RAID	**R**edundant **A**rray of **I**ndependent **D**isks
RAM	**R**andom **A**ccess **M**emory
RL	**R**einforcement **L**earning
SARSA	**S**tate **A**ction **R**eward **S**tate **A**ction
SLI	**S**calable **L**ink **I**nterface
SM	**S**treaming **M**ultiprocessor
SQL	**S**tructured **Q**uery **L**anguage
SSV	**S**tate **S**ignature **V**ector
TB	**T**era **B**yte
TRTH	**T**homson **R**euters **T**ick **H**istory
UQ	**U**pper **Q**uartile
VWAP	**V**olume **W**eighted **A**verage **P**rice

Chapter 1

Introduction and overview

Algorithmic trade execution considers the problem of optimally splitting a large order into multiple child orders to achieve some cost objective in financial markets, usually minimising cost with respect to a static or dynamic price benchmark. Human traders have traditionally performed this task, using a cultivated intuition and subtle understanding of market dynamics to optimally place child orders. As we turn to machines to replace this complex task at higher frequency time scales, we need to consider the validity of encoding the human trader's intuition and instincts into a rule-based system for machines, or whether a new perspective, the *machine's* perspective, should be cultivated to operate at this non-human scale. Indeed, a blend of perspectives is likely to be effective, but it is unclear how this can be optimally achieved. This thesis aims to address this issue by developing a coherent framework for automated, adaptive trade execution. To appreciate the machine's perspective, one must first establish a concrete paradigm for the nature of the system, a modelling framework which suits this paradigm, the governing dynamics of essential quantities, the visibility of measurable quantities for calibration, and a feasible scheme for online learning and adaptation.

1.1 The financial market as a complex adaptive system

Neoclassical economics, based on work developed by Jevons [143], Menger [183] and Walras [241], is one of the most influential theories to inform the mathematical modelling of modern economies. The central premise discusses economic agents as rational optimisers with perfect information, leading to general equilibria for optimal agent behaviour. While this offers mathematical convenience and tractability for analysing aggregate behaviour, empirical observation of agent dynamics, interactions and aggregate

consequences appear to differ from expectations under this paradigm [149], suggesting an alternative approach may exist which is closer in spirit to empirical observations.

In a landmark paper in 1898, Thorstein Veblen questioned the validity of the premise that economic systems tend to equilibria, and hence the validity of the use of equilibrium thermodynamics for studying the economy [238]. Veblen challenged the view of his contemporary, Léon Walras [241], arguing that *evolution* is paramount to the system dynamics and no effective theory could fully describe change and development in the system [239]. Hans Föllmer provided the first formal analysis of interacting heterogeneous agents in economics, using particle systems from statistical mechanics to create an economic interpretation of the Ising model [86]. This paper was clearly ahead of its time, as the theory of complexity economics was not yet mature enough to appreciate the validity of spin glass models for modelling observed phenomena. Alan Kirman reinforced the notions of biology and evolution in complexity economics, using an observation of the asymmetric behaviour of ants when faced with a symmetric situation, as an analogue for understanding the same observed phenomenology amongst interacting agents in financial markets [151]. Carl Chiarella interrogated the well-known cobweb model of price dynamics [148] when the equilibrium is unstable, showing that time lags between production and expectations could lead to a chaotic regime [53]. Duncan Foley developed a theory for characterising the statistical equilibrium distribution of competitive agents over outcomes, rather than seeking to predict specific agent outcomes, as the Walrasian paradigm suggests, introducing the entropy-minimisation criterion as the organising principle for price determination [85, 217]. John Holland discussed the concept of economies as complex adaptive systems, which exhibit evolving structure where components adapt to changing surroundings [130]. His key premise is that economies do not have a single governing equation, but rather consists of many interacting parts with no central control, each participating in influencing an outcome and the actions of others [130]. He further stresses that computer-based models of the economy need to incorporate these aspects if we want to further enhance our understanding of governing dynamics.

While the field of complexity economics has had many notable contributions over the last few decades (see [217] for a comprehensive history of the field), a key contribution was the establishment of the Santa Fe Institute for focused study of complex systems. In particular, the view of the broader economy as an evolving, complex system (as opposed to the Neoclassical paradigm [15]) was further developed in a pioneering discussion amongst economists, physicists, computer scientists and biologists at the Santa Fe Institute in 1987 [14]. Here, teams lead by Philip W. Anderson and Kenneth J. Arrow considered the implications of new ideas emerging in the natural sciences and how they may facilitate new ways of thinking about economic problems [17]. The meeting

highlighted six key features of the economy that challenged the prevailing mathematical models of Neoclassical economics [17]:

- *Dispersed interaction:* The dynamics of the economy is determined by the interaction of many dispersed, heterogeneous agents acting in parallel. The action chosen by any given agent depends directly upon the anticipated actions of other agents in the system, and on the aggregate system state which all the agents co-create.
- *No global controller:* There are no global entities which control interaction, but rather controls are provided via mechanisms of competition and coordination among agents. Actions may be mediated by governing or regulatory authorities, but not controlled at the individual level. There is also no universal competitor, i.e. an agent which can exploit all available opportunities in the system.
- *Cross-cutting hierarchical organisation:* Agents within an economy may be categorised into different levels of organisation with typical interactions among levels. Behaviours, strategies and actions at lower levels may inform those at higher levels, however the true organisation is not necessarily hierarchical.
- *Continual adaptation:* The behaviours, strategies and chosen actions of the agents are continually revised as the individual agents gain experience. The broader system thus also adapts.
- *Perpetual novelty:* Niches are continually created by new markets, trends, technologies, behaviours and institutions. By exploiting a niche, a new niche may be created in turn. The result is perpetual novelty and the continual possibility for improvement.
- *Out-of-equilibrium dynamics:* Since new niches and opportunities are continually created, the economy typically does not reach a globally-optimal state, or at least a globally-optimal state does not persist for a significant time. Improvements to chosen trajectories are thus always possible and can be expected.

These principles are thus key to any chosen modelling framework if one chooses to interrogate the empirical quantities of an economic system under the complexity economics paradigm. While the principles outlined above consider the broader economy, the ideas translate to our system of interest, namely the *equity market* or stock market. The equity market itself represents a prime example of an observable complex adaptive system. Many heterogeneous adaptive agents, such as traders, portfolio managers, market makers and regulatory authorities, interact non-linearly over time with each other and the electronic exchange, allowing for the emergence of complex behaviours beyond that

expected based on intrinsic agent characteristics. Many authors have viewed equity markets through this lens, considering analogues with physical systems to formulate models which aid our understanding of observed phenomena (see [16, 18, 47, 132, 246] and the references therein).

Wilcox and Gebbie offer a compelling paradigm for financial markets as a coupled, multi-level complex system [246]. They propose a mechanism for bottom-up and top-down causation, with level-specific effective models governing actors and inter-level interaction via noise terms. Actors at each level perceive the system in a different way, which invalidates the use of hierarchies of *the same* effective model to capture system complexity. A key insight is the specification of bottom-up agents generating market microstructure, via candidate representative models which encode observed agent interactions, providing plausible aggregation into price levels observed by system actors. They use Ising models [137] to govern stock-level dynamics and Potts models [251] to govern shared information factors across the market. While these are the simplest representations which capture the features of the system at the lower levels, the overall proposition by Wilcox and Gebbie permits isolated study of alternative generative dynamics of equity market microstructure. The work in this thesis can be contextualised as offering an alternative view for understanding the dynamics of market microstructure, preserving the use of spin glass models for capturing complex system behaviour, but with a nuanced perspective on encoding stock-level and market information factors.

Recent technological advances, accelerated by a highly competitive industry, have allowed for the efficient generation, storage and retrieval of financial data at micro time scales, providing a rich record of the price formation process as a laboratory for intensive study. The field of market microstructure developed to study the characteristics and behaviours of financial system dynamics at this scale (see [26, 37, 102, 127, 168, 206] for a comprehensive discussion). In particular, as intraday trading and investment processes become increasingly automated, understanding the system dynamics at varying intraday *time* and *event* scales is critical for an efficient trajectory through the system to be mapped by participating agents.

1.2 Spin glass models for modelling complex system behaviour

In physics, spin glasses are systems with localised magnetic moments, whose interactions are characterised by *quenched disorder* and *frustration*, which have analogous observed behaviours in complex systems [77, 226, 227]. The modern theory of spin glasses was

first introduced by Edwards and Anderson [77], who proposed that the essential physics of spin glasses are fully captured by the competition between quenched ferromagnetic and antiferromagnetic interactions amongst localised moments (or spins) associated with objects or particles, rather than their microscopic properties [227]. They introduced the following Hamiltonian:

$$H = -\sum_{i,j} J_{ij}\delta(s_i, s_j) - h\sum_i s_i \tag{1.1}$$

where i is a site on a d-dimensional cubic lattice, s_i is a spin associated with an object at site i, h is an external magnetic field, J_{ij} is the coupling between the ith and j^{th} spins and $\delta(.)$ is the Dirac delta function [227]. For illustrative purposes, we will consider a simple Ising model [137], where spins s_i are restricted to either $+1$ or -1. The *disorder* is represented by the J_{ij} coupling term, and is *quenched* in the sense that it remains fixed for all feasible laboratory time scales. The competition between ferromagnetic and antiferromagnetic interactions in Equation 1.1 indicates the presence of *frustration*, i.e. no spin glass configuration can simultaneously satisfy all couplings. This implies that spin glasses do not contain a unique equilibrium state, but rather can occupy one of many *metastable states*, each with approximately the same energy separated by free energy barriers [228]. In the complexity economics view, the properties of *quenched disorder*, *frustration* and *metastable states* are analogous with the principles of *dispersed interaction*, *continual adaptation*, *no global controller* and *out-of-equilibrium dynamics* discussed in Section 1.1. Although no laboratory spin glass has an energy function which looks like Equation 1.1 [227], the Edwards and Anderson Hamiltonian is conjectured to be the simplest Hamiltonian which can accurately model real spin glasses and hence is an ideal departure point for modelling complex systems [227].

In particular, we will consider the Hamiltonian in Equation 1.1 in the context of a q-state Potts model [251], where object spins s_i can take on one of q possible states. We will build on the work of Blatt et al. [38, 39, 249] and Giada and Marsili [103, 172], where they use an analogy to the ferromagnetic Potts model at thermal equilibrium to derive an unsupervised clustering algorithm for objects in financial markets. By assigning a Potts spin variable to each object and introducing a short-range distance-dependent ferromagnetic interaction field J_{ij}, regions of aligned spins emerge, which are analogous to groups of objects in the same cluster, where *spin alignment* suggests *object homogeneity* [242]. This provides an unsupervised scheme for determining the object spin configuration which best matches the induced interaction field at thermal equilibrium, and hence an optimal cluster configuration of objects. In Chapter 6, this concept is translated to temporal objects in financial markets, where a feasible set of intraday temporal market states is determined for the trading scale of the agent, thus

encoding a market information factor which efficiently summarises multiple features found in a dense market data feed and is measurable online.

1.3 The Hawkes process as the governing process for microstructure events

Alan Hawkes introduced a class of multivariate point processes with a stochastic intensity vector, modelling event-occurrence clustering and dependence in a coupled system [120]. Initial applications used calibrated Hawkes processes to measure the conditional intensities of earthquakes and aftershocks, based on recorded data [204, 205, 240]. In financial markets, empirical studies of market microstructure have highlighted apparent clustering of limit order book events at tick scale, with some event intensities exhibiting dependent behaviour [4, 36, 109]. Bacry et al. provide a comprehensive review article highlighting the many applications of Hawkes processes in finance [24]. Bowsher considered one of the first applications, where a bivariate point process of the timing of a stock's trade price and mid-quote changes was used to model volatility clustering on the New York Stock Exchange [46]. A key phenomenon investigated using Hawkes processes is *endogeneity* in financial markets [83, 84, 112, 113]. Empirical observation reveals that, in certain instances, market prices change too quickly to be strictly attributed to the flow of pertinent information, and thus evade explanation in classic economic theory [46]. By considering the ratio of exogenous parent events to endogenous events, it is possible to obtain a measure of market reflexivity [83].

Large used a multivariate Hawkes process to quantify the *resiliency* of a limit order book, viz. the propensity for quote replenishment following a liquidity demand event [159]. By characterising and extracting key liquidity demand and replenishment events from a limit order book, and using an appropriate choice of kernel to encode temporal dependence of events, Large claims it is possible to use a calibrated Hawkes process to calculate the probability and expected half-life of quote replenishment following a liquidity demand event [159]. We consider a similar approach, but instead use volume-conditional liquidity demand events. We identify key aggressive liquidity demand and replenishment events, enumerating empirical event point processes for model calibration. We investigate an appropriate kernel which provides a significant fit to the empirical data, before calibrating the multivariate Hawkes process to identify the resiliency of the order book with respect to market orders of varying size. Given that we need to measure the effects of our trading agent's interactions with the system, and the debate around accurately measuring permanent or transient price impact [42], we rather choose to use the resiliency study to inform the actions chosen by the trading agent. The actions, in

this case *market order size*, are constrained such that the order book is expected to be resilient with respect to the maximum permissible trade submitted by the agent. This is done by investigating the branching ratios of quote replenishment intensities following trade events of varying size. This ensures that an appropriate action set is determined for the stock concerned, such that the assumption of exogenous evolution of order book dynamics, for state representation purposes, is validated.

1.4 Reinforcement learning as a framework for online, adaptive trajectories through the complex system

Reinforcement learning is a technique used to numerically solve for a calibrated policy mapping states to optimal or near-optimal actions, given an objective in a stochastic system with unknown dynamics. Each state is a vector of observable attributes which describe the current configuration of the system. The technique proposes a simple, model-free mechanism for agents to learn how to act optimally in a controlled Markovian domain, where the quality of action chosen is successively improved for a given state using feedback from the system [67]. While the mathematical treatment of reinforcement learning evolved from the solutions to optimal control problems using value functions and dynamic programming [230], the conceptual foundations can be traced to the pioneering psychological studies of Edward L. Thorndike from 1896 to 1901 on the nature of animal intelligence [231]. Based on a series of behavioural conditioning experiments on different animals, he proposed the following two laws which govern acquired behaviour or learning:

> *Law of Effect: "Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond."* Edward L. Thorndike, p.244, 1911 [230, 231]

> *Law of Exercise: "Any response to a situation will, other things being equal, be more strongly connected with the situation in proportion to the number of times it has been connected with that situation and to the average vigor and duration of the connections."* Edward L. Thorndike, p.244, 1911 [231]

It is clear that these insightful conclusions inspired the notion of a learning algorithm which can converge towards favourable actions in different situations, simply based on feedback from the system of interest, permitting the possibility of developing intelligent purposeful computational agents. Subsequent work on the mathematical and computational advances of reinforcement learning theory should not lose sight of these basic behavioural foundations which govern learning. For our trading agent, we require a state representation which permits Markovian dynamics, a set of actions which yield sufficient and immediate feedback from the system for policy learning and some sense of the consequences of our actions on immediate state evolution. These aspects are considered in each of Chapters 5, 6, 7 and 8.

While reinforcement learning offers a compelling paradigm for deriving adaptive trajectories under Markovian dynamics, it rests firmly on the foundations of dynamic programming [29], assuming the system of interest is stationary and ergodic and hence guarantees convergence to a *single* globally-optimal state-action policy. If the system of interest is a complex adaptive system, the existence of a single, fixed-point solution for our objective should be questioned. Rather, reinforcement learning may still be effective if it converges to a *useful* policy *fast enough*, i.e. it learns at a rate faster than the natural time-scale of the system, yet adapts as new niches form. Galla and Farmer investigate the nature of agent learning by simulating two-person complicated games with varying payoff correlations, using experience weighted attraction (EWA) to evaluate the propensity for asymptotic learning in the parameter space [95]. They show that even under these simple conditions, chaotic regimes exist in the payoff correlation-learning rate space, where any attempts to learn a useful policy are inherently random. Their results suggest that, given a payoff correlation amongst competing agents, a learning rate should be chosen which ensures the agent is in a *fixed point* or *multiple fixed point* regime, such that feasible learning toward a useful policy is possible. Given the complex system paradigm we are considering, we would at best expect a (possibly dynamic) *multiple fixed point* regime to exist, thus a learning rate should be chosen which ensures we are within this region. It is unclear how the results presented by Galla and Farmer [95] can be extrapolated for real-world games, but it is clear that the chosen learning rate is critical for an effective trading agent and warrants careful evaluation for our study.

1.5 Automated algorithmic trading in modern financial markets

Barry Johnson defines *algorithmic trading* as a computerised rule-based system responsible for executing orders to buy or sell a given asset, consistent with the overall intention of our work [144]. As suggested by Johnson, it is prudent to clarify some further terminology at this point, as many terms referring to algorithmic trading are used interchangeably without an appreciation for the subtle nuances. *Portfolio trading* or *program trading* refers to a cost-effective means for trading multiple assets, using the economies of scale offered by a sell-side broker to significantly, yet optimally, alter the holdings of an investor's portfolio. *Systematic trading* refers to the case where the same approach is consistently adopted whenever a pre-encoded situation arises. While there may be a wide variety of rules for many possible situations, this paradigm does not permit adaptation. *Quantitative trading* moves beyond the realm of pure execution, where proprietary models may be used to initiate a trading decision (in algorithmic trading, the trading decision is often exogenous to the algorithm and the focus is on optimal execution). *High frequency trading* or *low latency trading* refers to a special case of quantitative trading, where trading agents take advantage of intraday opportunities, operating on time scales ranging from hours to fractions of seconds, or even *event time* [76]. *Statistical arbitrage* refers to a systematic trading approach which incorporates real-time data and historical analysis for trading or execution decisions.

In this thesis, we principally aim to develop an automated algorithmic trading agent which can learn to trade optimally intraday, given a trading decision with an arrival-price benchmark and finite trade horizon. In Chapter 9, we further introduce an approach which is conducive to automated high frequency quantitative trading, where the trading agent can learn to make decisions which maximise wealth and adapt as regimes change.

Although the techniques developed in this thesis are market agnostic, we are particularly interested in studying the South African equity market, accessed via the Johannesburg Stock Exchange (JSE). There have been a number of recent significant infrastructural changes which facilitate the deployment of automated algorithmic trading strategies, thus a rigorous study of changing market dynamics and development of bespoke trading agents will provide a meaningful and relevant contribution. In July 2012, the JSE's matching engine was physically moved from London, UK, to Johannesburg, South Africa (where the exchange resides), which reduced the round-trip trading latency by 400 times [184, 194], making lower-latency trading feasible. In September 2013, the cost model for execution fees was altered, whereby the prior cost floor for execution fees was removed in favour of a purely value-based system with increased cost ceiling [115, 195, 196, 199].

Prior to this change, small value trades were discouraged and penalised, whereas the fees now paid to the exchange are commensurate with trade size. This ensures that algorithms which split large orders into multiple child orders are now potentially profitable and effective. In May 2014, the exchange launched a colocation service, whereby clients can host their algorithmic trading engines directly at the exchange in leased rack space, further reducing round-trip trading latency to 100μs [197, 198]. These changes have fundamentally altered the market microstructure in the South African equity market, providing a rich landscape for studying the evolution of the financial system with each change. It also reinforces the need for adaptive trading algorithms, as the market dynamics are likely to be unstable in the near future as current participants and new entrants adjust their strategies in the new regime.

1.6 Overview

The remainder of this thesis is structured as follows: Chapter 2 contextualises the trade execution problem within the broader field of market microstructure, highlighting the nuances of interacting with the system at this scale. Chapter 3 discusses model-free reinforcement learning as a paradigm for finding adaptive trajectories in this system. Chapter 4 discusses the data used for this work, the choices which enabled efficient storage, retrieval and manipulation of large tick data volumes and a preliminary exploratory data analysis study revealing pertinent empirical details for modelling decisions. Chapter 5 discusses a simple model-free reinforcement learning algorithm for optimal trade execution, with a simplified discrete state representation and discrete actions. This algorithm demonstrates that reinforcement learning can be used to adapt a static arrival-price benchmark trading trajectory with respect to prevailing limit order book dynamics, significantly improving *ex post* implementation shortfall compared to standard models. Chapter 6 focuses on the refinement of the state representation for the reinforcement learning agent, using the complex system ideology to inform a unique approach for extracting persistent temporal dynamics from a streaming market data feed, to facilitate online learning. Chapter 7 focuses on the refinement of agent actions, using a calibrated volume-conditional multivariate Hawkes process to ensure that the limit order book is resilient with respect to the agent's interactions with the system. We examine the branching ratios of quote replenishment intensities related to specific trade events to identify the trade size where there does not appear to be commensurate quote replenishment. Chapter 8 incorporates the enhanced state representation and constrained actions into the reinforcement learning algorithm and discusses its efficacy compared to the simple model discussed in Chapter 5. Chapter 9 introduces a scheme

for online, unsupervised state discovery, detection and learning in high frequency markets, removing the need for human specification and pre-processing of state attributes, allowing the learning agent to find persistent structure in a streaming market data feed, enumerate its state space and learn to act optimally, at the scale of interaction. Chapter 10 summarises the key findings and provides some avenues for further research.

Chapter 2

Market microstructure and the trade execution problem

2.1 Overview

This chapter introduces the field of market microstructure, contextualising the trade execution problem and highlighting key features of our system of interest: the *limit order book*. For our trading agent to be effective, we need to understand the mechanisms and rules which govern the system at the micro scale, the consequences of interactions with the system, permissible trade-offs or compromises for the trade execution objective and the nature of observable attributes for calibration.

2.2 Market microstructure

In one of the major contributions of Léon Walras in 1874, *Éléments d'Économie Politique Pure ou Théorie de la Richesse Sociale*, he provided the first concrete discussion of the process of asset exchange between multiple parties, market frictions and price formation on the Paris Bourse, foreshadowing the field now known as *market microstructure* [241]. Mark Garman is credited with coining the term *market microstructure* in his 1976 paper of the same name, where he discussed the temporal dynamics of the transaction-to-transaction behaviour of prices, volumes, dealer inventories and market states [97]. Maureen O'Hara wrote one of the definitive texts on the subject, describing it as the study of the process and outcomes of exchanging assets under explicit trading rules [206]. Madhaven describes it as an area of finance that studies the process by which investors' latent demands are ultimately translated into prices and volumes [168]. A

comprehensive survey of the literature by Madhaven [168] and Biais et al. [37] reveals several major themes within the field of market microstructure:

- *Price formation and price discovery*: This theme primarily considers the manner in which information concerning an asset manifests in related price changes, as well as the consequences of trading on price dynamics. Initial work considered the role of market makers on the evolution of the bid-ask spread, initially as passive suppliers of liquidity [70], before considering the role of dealer inventory [97, 224] and asymmetric information [75, 107]. Hasbrouck proposed a novel approach which maps latent continuous models of prices and trading costs to observed bid-ask quotes which are discrete and clustered [118]. This work provided the precedent for using diffusion processes for modelling price evolution at the tick scale, even though observed prices appear to evolve as discrete jumps. French and Roll examined the intraday seasonality of stock return variance, where empirical observation confirmed higher return variance during trading hours [92]. They conjectured that this could be due to higher frequency of public information arrival during business hours, informed agents acting on private information during business hours and the process of trading itself creating volatility. Following this work, and as transaction-level data became more freely available, many authors investigated the causal effects of empirically observed price anomalies and patterns in observed quantities in financial markets [74, 114, 141, 178, 250]. Since we are concerned with the effects of agent interactions with the system, studies on price impact and order book resiliency are of particular interest. These are discussed in more detail in Sections 2.4 and 2.5.

- *Market structure and design issues*: This theme considers the impact of market architecture, regulation and design on market quality metrics, such as spreads, liquidity and volatility [168]. This may include the degree of continuity in trading (auctions versus continuous trading), choice of quote-driven or order-driven market, floor trading versus electronic trading, order types permitted on the exchange, regulatory protocols governing trading, availability of dark pools or anonymous crossing venues and the degree of transparency amongst different classes of market participants. Each of these will have direct consequences for the behaviour of trading agents, and hence influence price formation.

- *Information and disclosure*: This theme focuses on the effect of asymmetric information availability amongst market participants on price formation. In particular, it focuses on *market transparency*, i.e. the ability of market participants to observe information relating to the trading process [206]. Bloomfield and O'Hara find that low-transparency dealers are able to exploit informational advantages

and outperform high-transparency dealers, in terms of revenue [40, 41]. Other studies consider the role of anonymity in price formation [88] and disclosure of information concerning pending orders [7, 167].

- *Informational issues arising from the interface of market microstructure*: This theme considers the interface with asset pricing, such as the effect of liquidity on expected returns, as compensation for expected transaction costs, as well as the behavioural effects of traders who tend to be overly aggressive, due to overestimation of the precision of their information [168]. Other studies consider the effect of cross-border flows on price formation for stocks listed on multiple exchanges [111].

There is clearly a rich and multifaceted history of market microstructure analysis in modern financial markets and this brief summary highlights the many nuances which influence observed prices on electronic exchanges. As our laboratory of interest, the *limit order book* is the system which encapsulates these features and we will consider its mechanics in more detail, so as to better understand the drivers of its temporal state evolution.

2.3 The limit order book and trading mechanism

Gould et al. [109], Abergel et al. [4] and Jaimungal et al. [140] provide excellent references on the mechanics of limit order book markets, the trading mechanism, as well as the typical formalism for the mathematical modelling thereof. For completeness, we will provide a summary exposition here which highlights the nuances of the rules which govern price formation via trade execution in order-driven markets.

A limit order book (LOB) is a device used by electronic exchanges to collate the intentions of market participants via their submitted orders, where each order typically contains the sign (buy or sell), desired transaction price, desired quantity, timestamp and type. At a particular point in time, the LOB thus summarises the market's quoted intentions, whereas the time evolution of the LOB reflects the way the market reacts under the influences of its participants [4]. A trade takes place when there is a commensurate match of an existing limit order by a willing party on the other side of the transaction, and the sequence of matching is usually governed by a price/time priority algorithm. Typically, there are four types of orders which can be submitted to a LOB:

- *Limit order*: A quote which specifies the price and quantity at which a participant is willing to transact (buy or sell). *Ask quotes* are limit orders to sell a stock and *bid quotes* are limit orders to buy a stock. Ask quotes are prioritised from

lowest to highest price, whereas bid quotes are prioritised from highest to lowest price. If two quotes on the same side of the book have the same price, they are prioritised by arrival time to the LOB. The lowest ask quote price is referred to as the *best ask* and the highest bid quote price is referred to as the *best bid*. The difference between the *best ask* and *best bid* price is referred to as the *spread* and the arithmetic average of the *best ask* and *best bid* price is referred to as the *mid-price*. All the quotes on the bid and ask side of the order book are organised into numbered *levels*, with lower levels indicating closeness to *best bid* and *best ask* quotes, making up market depth. For limit orders, there is no guarantee of execution - a full or partial match may take place if a willing counterparty to the transaction arrives at the LOB. A limit order thus offers the possibility of transacting at a favourable price, at the expense of execution uncertainty. The complete schedule of active limit orders makes up the LOB at a given time.

- *Market order*: An order to immediately buy or sell a given quantity of shares at the best available price. For a buy (sell) market order, this will match the commensurate number of sell (buy) limit orders in *market depth* level sequence, resulting in an elevated (depressed) execution price based on the number of ask (bid) quote levels matched. The execution price is calculated based on the volume-weighted average price of the matched quotes. The market order thus offers execution certainty, at the expense of the transaction price level attained.

- *Modification*: This is an order which modifies the price level, quantity or type of an existing active limit order quote. Modifying an existing quote preserves its time priority.

- *Cancellation*: This is an order which cancels an existing active limit order quote.

Figure 2.1 illustrates the process of liquidity demand and replenishment in a LOB [4]. *Bid quotes* are shown in lighter grey and *ask quotes* in darker grey. In each subfigure, the vertical axis shows *signed quantity* and horizontal axis shows the *price level* of each prevailing *limit order quote*. Consider a trading agent faced with the initial schedule of limit orders shown in (1), with the intention of *buying* a certain quantity of the asset. The agent chooses to execute a *market order*, fully matching levels 1 and 2 of the *ask quote depth* and partially matching level 3, obtaining a transaction price equivalent to the volume-weighted average of the price levels of the matched quotes. We note that the transaction price is necessarily higher than the prevailing *mid-price*. The difference between the transaction price and the mid-price can be thought of as the cost of certain, immediate execution. The matched quotes are removed from the LOB, creating a transient deficit, widening the *spread* and increasing the *mid-price*, as shown

in (3). This short-term change in the *mid-price* as a result of trading is also referred to as *temporary price impact*, discussed in Section 2.4 below. Other market participants then submit new limit orders to the LOB, restoring liquidity, as shown in (4), (5) and (6). The speed and extent of limit order quote replenishment following a trade event, shown in (4) and (5), is also referred to as LOB *resiliency*, discussed in Section 2.5 below.

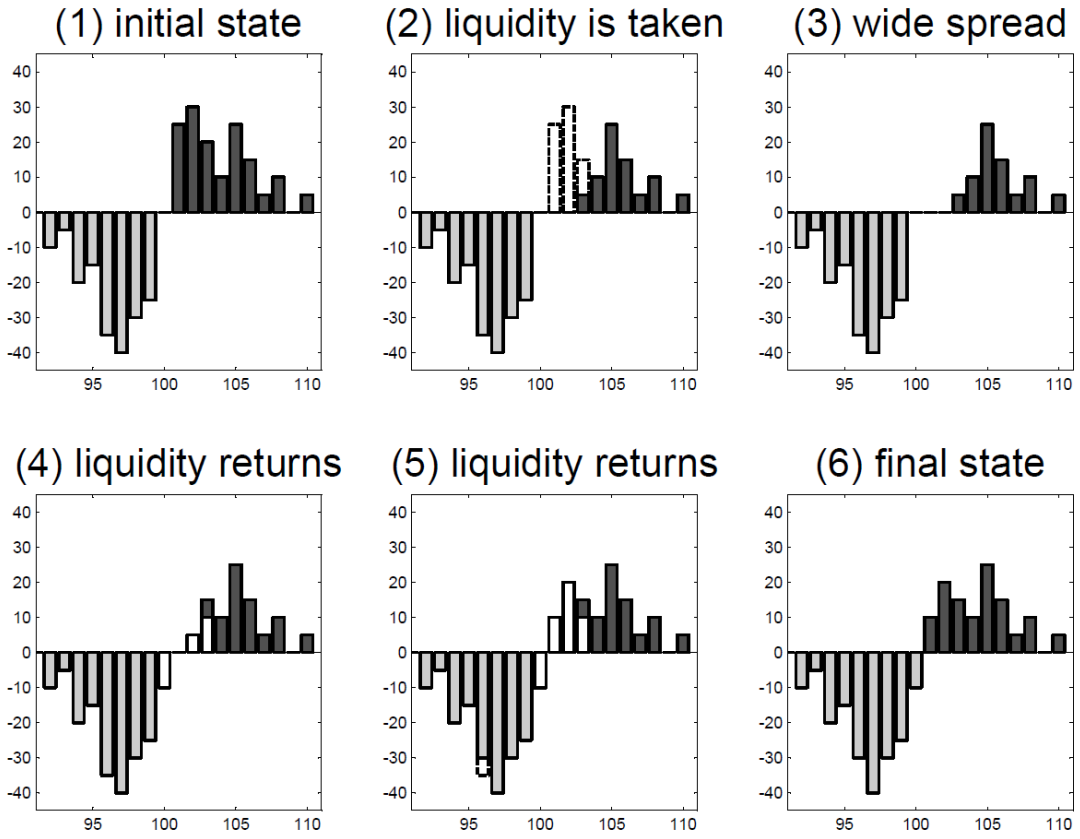


FIGURE 2.1: Illustrating the effect of liquidity demand and subsequent replenishment. A buy *market order* arrives and removes commensurate quotes from the ask side of the limit order book (2), creating a transient deficit (3). Bid and ask limit orders are then submitted by other participants (4,5) and liquidity is restored (6). This figure is reproduced from the text, *Limit Order Books* by Abergel et al. [4].

Figure 2.2 further illustrates some stylised aspects of a snapshot of the LOB at a particular point in time. Here, green lines indicate *ask quotes* and blue lines indicate *bid quotes*, with the vertical position showing the price level and line length showing quote quantity. It is clear the larger market orders which exhaust more limit order quotes *push the price*, i.e. there are adverse consequences for the resultant transaction price and temporary consequences for the mid-quote price and spread. The instantaneous distribution of market depth quotes also has a direct impact on trading. Here we see sparsely spread bid quotes with lower associated quantities than ask quotes, suggesting that sell market orders would push the price further (in absolute terms) than buy market orders of the same quantity. The ratio of ask quotes to bid quotes is often referred to

as *order imbalance*, and has been considered as an indicator of short-term price moves [50].

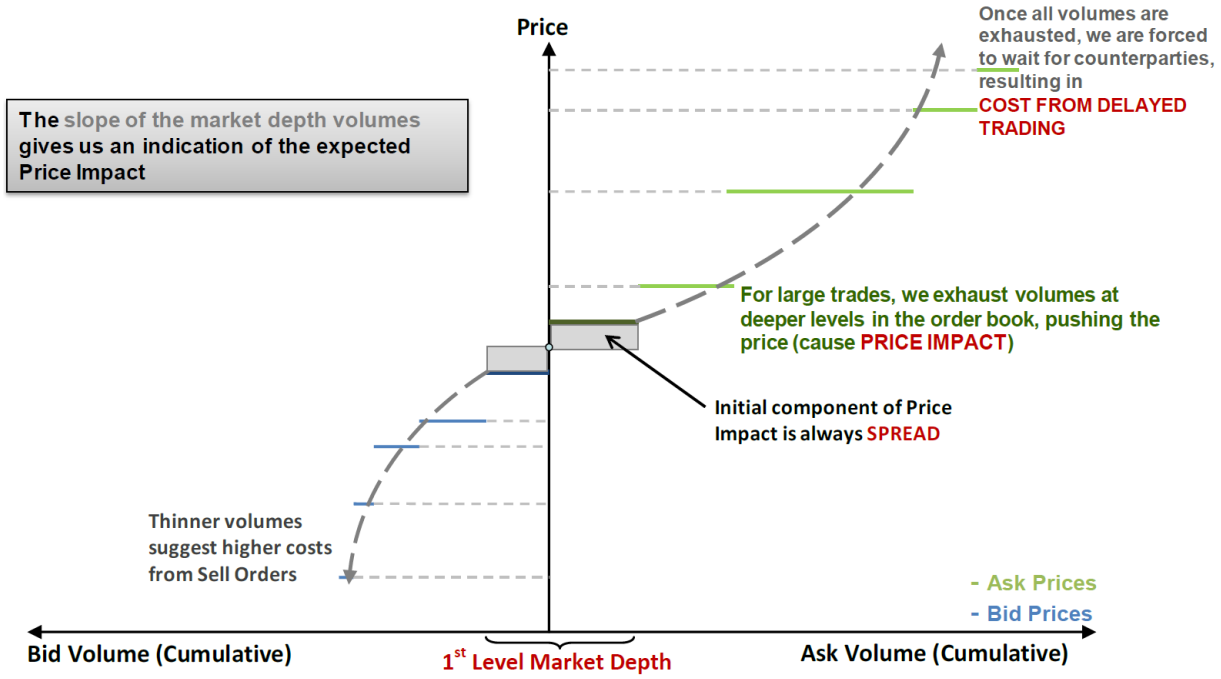


FIGURE 2.2: Some stylised aspects of a limit order book. Green lines indicate *ask quotes* and blue lines indicate *bid quotes*, with the vertical position showing the price level and line length showing quote quantity. This figure illustrates market depth, quote imbalance and temporary price impact.

2.4 Price impact

Price impact broadly refers to the consequences of an executed order to buy or sell a certain quantity of an asset on the subsequent price dynamics for that asset. In the chapters which follow, we will be assuming that our trading agent will be executing *market orders* based on the prevailing LOB, using the order quantity as the control when navigating the system. It thus seems prudent to consider the effect of market orders of varying size on LOB and price dynamics, since this has a direct impact on the chosen control.

Several studies have investigated various aspects of price response to trade events [42, 43, 74, 82, 98, 115, 136, 164, 186, 210, 212, 248, 253]. Initial studies considered the mid-price as the fundamental price of a stock, quantifying the average change in *log mid-quote* price as a function of *traded volume* [164]. This permitted the calibration of *price impact*

curves, which provide an indication of the expected cost of trading various quantities of a share, beyond any direct costs such as exchange fees or broker commissions. Since these studies focus on the change in mid-quote price immediately before and after a trade, these curves, which typically exhibit concave volume dependence [42], appear to capture *temporary price impact*, i.e. that aspect of the elevated/depressed transaction price that is expected to dissipate by the next trading opportunity. This may be caused by a market order absorbing limit order quotes beyond level 1, resulting in a price deviation which exceeds the half-spread, however the ensuing replenishment of quotes ensures that the effect is corrected. Certain transactions may cause price effects which persist to subsequent trading periods, either by revealing investment intentions causing other market participants to respond, or depleting liquidity in the LOB without commensurate quote replenishment. This effect is referred to as *permanent price impact*, since it affects the evolution of the fundamental price.

Bouchaud et al. studied a large number of transactions in a variety of markets, and found that the autocorrelation of the signs of trades decays very slowly with time, suggesting order flow is strongly persistent and predictable [42, 44]. This suggests that private information is incorporated into fundamental prices at a much slower rate than suggested under the permanent impact paradigm, which assumes uncorrelated trade signs. They thus introduce a *transient price impact* model, which uses a kernel function with a decaying shape offsetting the autocorrelation of order flow, to model the slower manifestation of private information in fundamental prices [42].

It is clear that the trading consequences of our agent's actions will affect both the feedback received from the system, in the form of the transaction price versus the arrival price, as well as the evolution of the state space, via transient and permanent price effects, consistent with the complex adaptive system paradigm. It also appears that price impact and LOB quote replenishment are inextricably linked, and we will explore this connection further in Chapter 7 as a means to control our agent's actions.

2.5 Order book resiliency

Resiliency can be defined as the propensity for limit order quote replenishment following a liquidity demand event, i.e. the speed and extent of reversion of the LOB shape to its form prior to the trade event, as demonstrated in Figure 2.1. While Kyle is credited with introducing the concept in his description of liquidity [156], Large formalises resiliency using continuous-time impulse response functions based on conditional order book event arrival intensities, modelled as a multivariate Hawkes point process [159]. This model is calibrated using LOB data from the London Stock Exchange (LSE). He found that

the LOB failed to replenish resiliently with a probability of at least 60%. In particular, when considering resiliency events that apply only to the bid or ask side of the order book, he finds that resilient replenishment after a large market order fails more than 80% of the time. This result is also seen in other key studies by Danielsson and Payne [64] and Degryse et al. [69]. Since Large used an exponential kernel to encode temporal dependence of events, he further quantified the average half-life of resilient replenishment to be under 20 seconds. This implies that if the order book does replenish, it has more than 50% chance of doing so within 20 seconds after a liquidity demand shock [159].

Degryse et al. consider the resiliency of a limit order book using non-parametric techniques, analysing order flow around aggressive orders [69]. They use the approach of Biais et al. to classify the *aggressiveness* of a buy or sell order [36]. Resiliency is analysed by investigating how bid and ask prices, spreads, depth, duration, order flow and transaction prices behave over a specified time window subsequent to the event. They confirm the so-called *diagonal effect* of serial correlation reported by Biais et al., demonstrating that it takes approximately 50 subsequent orders before the order flow returns to its unconditional pattern following an aggressive order [36]. Degryse et al. further find that the spread initially increases significantly above its average following an aggressive order, but reverts to normal levels within 20 level-1 quote updates of the LOB [69].

They further find that the initial price impact caused by a transaction is partially reversed in subsequent transactions, but that there are long-term effects from aggressive orders. Thus, the average transaction prices of buys (sells) after an aggressive market order are higher (lower) than pre-event transaction prices [69]. In a study on the South African equity market, following an observed increase in average price impact for low-volume trades as a result of exchange fee restructuring [115], Hendricks and Harvey propose a potential LOB resiliency explanation. They show that an increase in the intensity of low-volume trades without commensurate quote replenishment resulted in elevated price impact for trades below a certain size [123]. This demonstrates the link between price impact and resiliency, and will be investigated further in Chapter 7 as a means to control our trading agent's actions.

Daniellson and Payne examine how liquidity is determined on a limit order book for foreign exchange trade, the Reuters D2000-2 [64]. They introduce the concept of what they call *dynamic* liquidity or illiquidity. This aims to capture what occurs after a market order arrives at the LOB. A LOB is considered to be *dynamically illiquid* if a market buy (sell) causes a *further* removal of liquidity from the sell (buy) side, i.e. the response to a liquidity demand shock is to *not* supply new liquidity around the same price. They find that the foreign exchange limit order book studied is dynamically illiquid (not resilient).

They further find that if repetitive intraday patterns are controlled for, the buy and sell side depth are uncorrelated. This is similar to Large's finding, where replenishment is equally likely to occur on either side of the LOB [159].

2.6 Optimal trade execution

A critical problem faced by participants in financial markets is the so-called *optimal trade execution* problem, viz. how *best* to trade a given block of shares to achieve *minimal cost*. Here, cost can be interpreted as in Andre Perold's *implementation shortfall* [209], i.e. adverse deviations of actual transaction prices from an arrival price *frictionless* baseline when the investment decision is made. Alternatively, cost can be measured as a deviation from the market volume-weighted trading price (VWAP) over the trading period, effectively comparing the specific trader's performance to that of the average market trader. In each case, the primary problem faced by the trader/execution algorithm is the compromise between price impact and opportunity cost when executing an order.

As discussed in Section 2.4, price impact here refers to adverse price moves due to a large market order absorbing limit order quotes at available levels in the LOB, pushing the price and causing *temporary price impact*. As market participants begin to detect the total volume being traded and the trader's intentions, they may also adjust the price level of their submitted quotes downward/upward to anticipate order matching, resulting in *permanent price impact* [131]. To avoid price impact, traders may split a large order into smaller child orders over a longer period. However, there may be exogenous market forces which result in execution at adverse prices, i.e. an *opportunity cost* from delayed trading. This behaviour of institutional investors was empirically demonstrated in [52], where they observed that typical trades of large investment management firms are almost always broken up into smaller trades and executed over the course of a day or several days. A heuristic solution may involve choosing the maximum permissible child order trade size such that the expected temporary impact is less than the measured price volatility for the duration of the trading program, but the problem quickly becomes more complex.

This concept is illustrated in Figure 2.3, which will serve as the primary problem description which we aim to solve in this thesis. Consider a trading agent which receives an instruction to sell 60 000 shares of some stock at 09:55. The trading agent receives further instructions to guarantee execution by 10:30 and aim to match the *arrival price* (minimise implementation shortfall). Here, the *arrival price* is the prevailing price level

at the time the trading agent receives the instruction to trade, indicated by the red dotted line. The grey bars indicate the market's total traded volumes in successive 5-minute periods. The blue bars indicate the trading agent's chosen participation in traded volumes, i.e. each represents a market order with the quantity shown by the blue bar and the price attained is approximated by the coincident level of the black line. The orange line then shows the running VWAP of the trading agent's order, based on the executed volumes and price levels. Once the entire quantity is sold (at time 10:25), the trading agent's achieved VWAP is compared to the arrival price of the trading programme. In this case, the trading agent has created value (achieved a sale price higher than the arrival price) directly as a function of the chosen quantities and timing of market orders executed over the trading program. We note some terminology at this point: an *acquisition* is a trading program to *buy* a specified quantity of shares and a *liquidation* is a trading program to *sell* a quantity of shares. These are of course special cases of the general *optimal trade execution* problem.

We aim to develop a trading agent which optimally chooses these market order volumes, however we will consider a variety of calendar and event time scales governing the frequency of execution, carefully considering an appropriate state representation which permits learning.

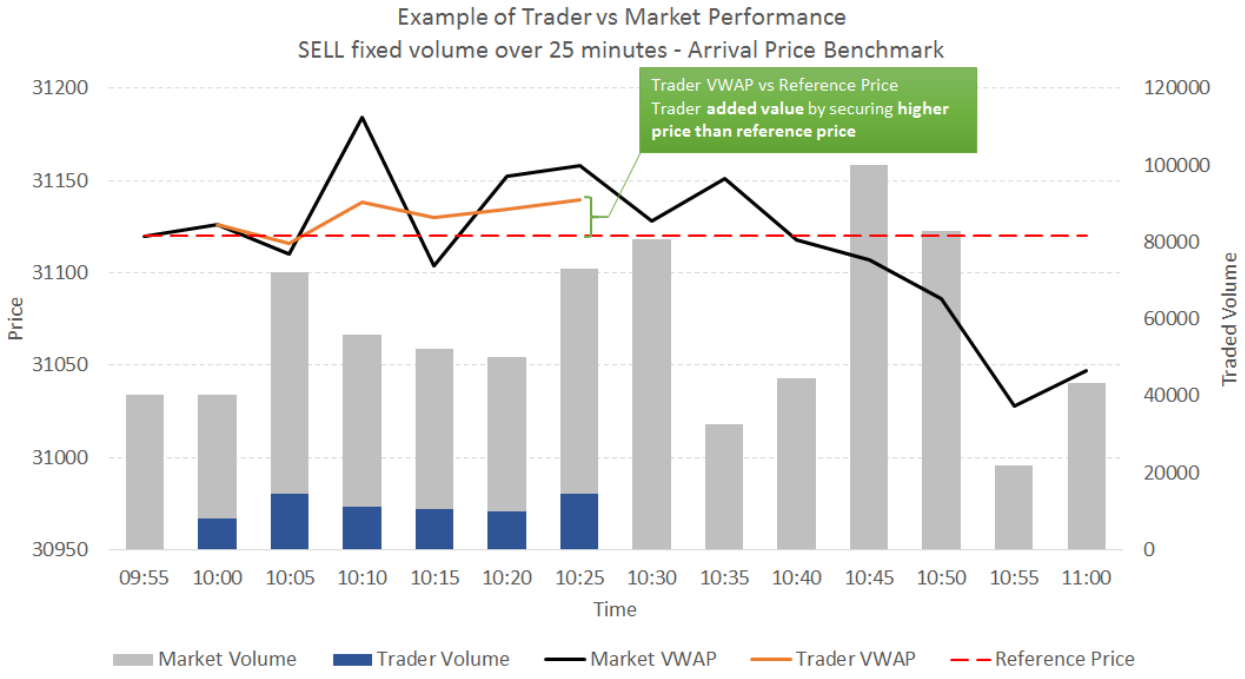


FIGURE 2.3: Trade execution with arrival price benchmark

The fine resolution evolution of the LOB as our trading agent's state space, the extraction of persistent features at the trading scale which can be exploited for optimal execution and the effect of the agent's interactions with the system on its evolution are all key

considerations for the development of an unsupervised trading agent which can adapt online.

2.7 Some remarks

The are some key computational challenges which should be noted when interacting with this system at this scale. The LOB data is by nature asynchronous with varying levels of throughput, i.e. each time a new limit order, modification, cancellation or market order is submitted, the associated event is recorded with a time-stamp. We have access to recorded market depth at micro-second resolution, thus the sheer volume of data quickly moves beyond the typical memory capabilities of desktop computing solutions. This places this work firmly in the *big data in finance* domain, requiring bespoke high-performance computing solutions for practical efficacy. In addition, traditional time-series analysis techniques for multiple objects assume synchronous arrival times of data. This assumption requires revision if we aim to extract inferences at this micro scale without imposing any biases. In particular, for an online trading agent, we need to ensure that it can make sense of a streaming market data feed of asynchronous events.

The drivers of the temporal evolution of LOB dynamics are much more complex than indicated in this chapter. Beyond pure intentions of buying or selling an asset based on an investment decision, trading agents employ various tactics to manipulate LOB dynamics to achieve favourable transaction costs or extract alpha [119, 181]. *Quote stuffing* is a practice where a high-frequency trader operating in a market with multiple trading venues floods a particular venue with worthless quotes, creating a *stale* representation of the market's intentions, before using their technological advantage to react on another trading venue faster than other participants. *Quote spoofing* involves creating false demand by loading up one side of the LOB with quotes, falsely signalling a buy/sell intention to participants, such that a particular limit order has a higher chance of being matched. The deception of *spoofing* undermines the ability of value investors to identify fundamental value of the assets being traded. These practices cause significant quote volatility, making the extraction of useful, persistent signals more difficult. Any paradigm used to model dynamics at this scale needs to be robust with respect to the enumerable drivers of observed behaviour.

Chapter 3

Model-free reinforcement learning

3.1 Overview

Reinforcement learning (RL) is a technique used to numerically solve for a calibrated policy mapping states to optimal or near-optimal actions, given an objective in a stochastic system. Each state is a vector of observable attributes which describe the current configuration of the system. The technique proposes a simple, model-free mechanism for agents to learn how to act optimally in a controlled Markovian domain, where the quality of action chosen is successively improved for a given state using feedback from the system [67].

The problem of solving for an optimal policy mapping states to actions is well-known in optimal control theory, with a significant contribution by Bellman [29]. Bellman showed that the computational burden of a Markov Decision Process (MDP) can be significantly reduced using what is now known as dynamic programming. It was however recognised that two significant drawbacks exist for classical dynamic programming: Firstly, it assumes that a complete, known model of the environment exists, which is often not realistically obtainable. Secondly, problems rapidly become computationally intractable as the number of state variables increases, and hence, the size of the state space for which the value function must be computed increases. This problem is referred to as the *curse of dimensionality* [230].

RL offers two advantages over classical dynamic programming: Firstly, agents learn online and continually adapt while performing the given task. Secondly, the methods can employ function approximation algorithms to represent their knowledge. This allows generalisability across the state space, improving learning time and exploration properties of the algorithm [72]. Reinforcement learning algorithms do not require knowledge

about the exact model governing an MDP and thus can be applied to MDP's where exact methods become infeasible.

Although a number of implementations of RL exist, we will focus on *Q-learning*. This is a model-free technique first introduced by Watkins [243], which can be used to find the optimal, or near-optimal, action-selection policy for a given MDP. In the context of the optimal liquidation problem, the algorithm can be used to examine the salient features of the current order book and current state of execution in order to decide which action (e.g. child order price or volume) to select to service the ultimate goal of achieving the execution objective.

3.2 Markov Decision Processes

To understand the current theory for RL, one must first consider the formalism of discrete-time controlled Markov Decision Processes (MDPs) as a model for temporal system evolution. A number of notable resources on MDPs are available (see for example [30, 213, 214, 216, 245]). In particular, White and White provide a comprehensive review of MDPs [245], and a summary exposition will be presented here to facilitate development and context for the RL approach to optimal control.

An MDP is a mathematically-based optimisation model of discrete-stage, sequential decision making in a stochastic system [245]. It consists of four primary components: *states*, *actions*, *rewards* and *transition probabilities*. A learning agent is able to perceive the current *state* of the world, which encodes all observable features describing the system configuration. The agent can then perform an *action*, which yields some immediate feedback (*reward*) and causes the system to evolve to the next *state*. Since the system is stochastic, the *state* evolution consequences of a particular *action* is not deterministic, thus there is a probability distribution over the possible states at the next decision step (*transition probabilities*). The characteristic feature of an MDP is that the transition probability from the current *state* to the *state* at the next decision point is dependent only on the current *state*, and not earlier *states* or *actions* of the process. Equivalently, this means that all the information necessary for an agent to make an *action* decision is encoded by the current *state*. In this scenario the MDP is considered *controlled*, in the sense that the agent's chosen *action* directly affects the temporal *state* evolution in some way.

More formally, let $\{s_t : t = 1, 2, ..., T\}$ be a controlled Markov chain with finite horizon length, T. Assume an agent is able to perceive the current state $s_t \in \mathcal{S}$ and can choose a candidate action to perform, $a_t \in \mathcal{A}$. Here, $\mathcal{S}$ is a compact subspace of $\mathbb{R}^p$ representing

the state space and $\mathcal{A}$ is a finite set of actions. The dynamics of $\{s_t : t = 1, 2, ..., T\}$ can be described by the following (time-independent) conditional probabilities:

$$Prob(s_{t+1} = j | s_t = i, a_t = a) = P_{i,j}(a). \tag{3.1}$$

$P_{i,j}(a)$ is thus a probability kernel encoding the probability of moving from some state $i \in \mathcal{S}$ to some state $j \in \mathcal{U}$ using control $a \in \mathcal{A}$, where $\mathcal{U}$ is a subset of $\mathcal{S}$ (i.e. $\mathcal{U} \subseteq \mathcal{S}$).

The function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as the reward function, such that $r(i, a)$ is the immediate numerical reward incurred when the agent performs action a in state i.

A key concept in the theory of MDPs is that of a *policy*, viz. a particular candidate mapping from *states* to *actions*. Mathematically, this is represented as:

$$\Pi(\mathcal{S}) = \{\pi(i) = a : i \in \mathcal{S} \text{ and } a \in \mathcal{A}\}. \tag{3.2}$$

Note, we are assuming that the policy is *stationary*, i.e. it will perform the same action a whenever it encounters state i, regardless of when this occurs in time. Given that some policy Π deterministically encodes an action for every state, and we know the reward consequences for such actions (r), we can ascribe a *value* to each state, which represents the expected cumulative reward incurred by following the action trajectories from Π until the end of the problem horizon or the terminal state is reached. This *value function* is defined as:

$$V^{\pi}(i) = r(i, \pi(i)) + \gamma \sum_j P_{ij}(\pi(i)) V^{\pi}(j) \quad \text{for } i, j \in \mathcal{S}. \tag{3.3}$$

Here, γ denotes the constant discount rate ($0 < \gamma < 1$), which ensures a preference for immediate rewards versus future rewards. In general, a larger γ places more weight on future rewards and can result in a more difficult optimisation problem [67].

The goal of the learning agent is to maximise the expected cumulative sum of discounted rewards, viz. find the policy Π^* such that

$$V^{\pi^*}(i) = \arg\max_{\pi} \{r(i, \pi(i)) + \gamma \sum_j P_{ij}(\pi(i)) V^{\pi}(j)\} \quad \text{for } i, j \in \mathcal{S}. \tag{3.4}$$

Equation 3.4 is a version of the Bellman optimality equation for value functions [29, 31]. Finding the optimal policy Π^* directly is a non-trivial task, even when system dynamics are known exactly. Since each action taken affects the temporal system evolution at subsequent time steps, every possible sequence of actions needs to be considered to evaluate the value function, which grows exponentially in sequence length. Bellman recognised this intractability and developed a set of methods for solving this optimisation

problem, utilising the Markov property to reduce the scale of the problem [29, 31]. This procedure is termed *dynamic programming*, which is the analytical precursor to RL theory.

3.3 Dynamic Programming

The dynamic programming optimisation approach exploits the structure inherent in certain classes of complex problems, which can often be broken up into a system of simpler, overlapping sub-problems which are easier to solve. The solution to the original problem is then reconstructed from the sub-problem solutions. MDPs are ideal candidates, since Equation 3.4 provides a recursive decomposition of the problem and the value function $V^{\pi}(i)$ stores and reuses its solutions for a particular policy Π. In particular, finite-horizon MDPs often have a trivial optimal terminal state mapping, $(\pi(s_T), s_T \in U \subset \mathcal{S})$, which can be computed and stored. The optimal mapping for $(\pi(s_{T-1}), s_{T-1} \in \mathcal{S})$ can then be solved, given that we know the optimal mapping at time T. Solving the set of sub-problems in this predefined order is referred to as *backward induction* and is appropriate for finite-horizon MDPs with stationary, deterministic policies.

There are two primary algorithms in dynamic programming to solve each of the sub-problems: *policy iteration* and *value iteration*.

3.3.1 Policy iteration

As described in Section 3.2, a *policy* is a particular mapping from states to actions. For this exposition, we consider only deterministic policies, however more generally policies can be stochastic, i.e. with some probability distribution over candidate actions for a particular state (see for example [81]). In addition, we assume policies are *stationary*.

Policy iteration broadly involves two stages: policy *evaluation* and policy *improvement*. Policy *evaluation* involves determining the discounted cumulative reward from following the exact trajectory of actions from a candidate policy in the current and subsequent states, i.e. evaluating Equation 3.3 for a given state i. Policy *improvement* uses the values $V^{\pi}(i)$ to specify a new policy Π' which is guaranteed to be at least as good as Π. We will omit the details here, but a key insight is the specification of a *Q-function*, which determines the value of a non-stationary policy whereby some action a is executed in the current state, and the policy mappings from Π are used in subsequent states, i.e.

$$Q^{\pi}(i,a) = r(i,a) + \gamma \sum_{j} P_{ij}(a) V^{\pi}(j) \quad \text{for } i,j \in \mathcal{S}. \tag{3.5}$$

The primary difference between Equation 3.3 and Equation 3.5 is that we allow the case were $a \neq \pi(i)$. A new policy is then chosen, such that

$$\pi'(i) = \arg\max_{a \in \mathcal{A}} \{Q^{\pi}(i,a)\} \quad \text{for } i \in \mathcal{S} \tag{3.6}$$

where it is clear that $Q^{\pi}(i,\pi'(i)) \geq Q^{\pi}(i,\pi(i))$ and the overall policy is thus improving simultaneously in every state $i \in \mathcal{S}$ [67]. If $\pi'(i) = \pi(i)\ \forall\ i \in \mathcal{S}$, then Π is an optimal policy, usually denoted by Π^*. The value function and *Q-function* associated with the optimal policy are denoted by $V^{\pi^*}(i)$ and $Q^{\pi^*}(i,a)$ respectively. Policy iteration is guaranteed to converge to an optimal solution, since the problem is finite and improves measurably at each time step [133].

3.3.2 Value iteration

Value iteration is an alternative proposition in dynamic programming theory to solve for the optimal policy. Under value iteration, the *Q-function* is updated directly according to the following equation

$$Q'(i,a) = r(i,a) + \gamma \sum_{j} P_{ij}(a) \arg\max_{b \in \mathcal{A}} Q(j,b) \quad \text{for } i,j \in \mathcal{S} \tag{3.7}$$

following an appropriate initialisation of Q. Although the *Q-function* as specified in Equation 3.7 is not associated with a particular policy, as in Equation 3.5, it can be shown that if Equation 3.7 is iterated infinitely-often, it will converge to the *Q-function* associated with the optimal policy [25, 29, 31]. The optimal policy is then defined as

$$\Pi^*(\mathcal{S}) = \{a : Q(i,a) = \arg\max_{b \in \mathcal{A}} Q(i,b) \text{ and } i \in \mathcal{S}\}. \tag{3.8}$$

We will omit the details of the proof here, but a key insight is that

$$||Q'(i,a) - Q^{\pi^*}(i,a)||_{\infty} \leq \gamma ||Q(i,a) - Q^{\pi^*}(i,a)||_{\infty} \quad \text{for } 0 < \gamma < 1 \tag{3.9}$$

viz., the *L_{∞}-norm* of the distance between $Q'(i,a)$ and $Q^{\pi^*}(i,a)$ is less than that between $Q(i,a)$ and $Q^{\pi^*}(i,a)$ by a factor of at least $0 < \gamma < 1$ at each iteration step [29, 31, 67]. The *Q-values* thus converge exponentially to their optimal values, ensuring the policy is at least near-optimal after a finite number of steps. The value iteration algorithm is closely related to a simple reinforcement learning algorithm, *Q-learning*, which considers the case where transition probabilities and rewards are at least partially unknown *apriori*, making dynamic programming unsuitable. This is the primary algorithm we will be considering and extending in this thesis.

3.4 The *Q-learning* algorithm

Q-learning is a model-free reinforcement learning technique first introduced by Watkins [243], which can be used to find the optimal, or near-optimal, action-selection policy for a given MDP [244]. It uses the idea of the Bellman optimality equation and the value iteration algorithm to derive a update rule which successively refines the state-action policy mapping using simple observed feedback from the system, without full knowledge of transition dynamics.

The task of the *Q-learning* agent is to determine Π^* where $P_{ij}(a)$ is unknown, using a combination of exploration and exploitation techniques over the given domain. In the exposition which follows, we will use $*$ to indicate the optimal policy π^* where this eases notation.

It can be shown that $V^*(i) = \max_a Q^*(i, a)$ and that an optimal policy can be formed such that $\pi^*(i) = a^*$. It thus follows that if the agent can find the optimal Q-values, the optimal action can be inferred for a given state i. It is shown in [243, 244] that an agent can learn Q-values via experiential learning, which takes place during sequential episodes. At the tth iteration, the agent:

- observes its current state S_t,
- selects and performs an action A_t,
- observes the subsequent state S_{t+1} as a result of performing action A_t,
- receives an immediate reward r_t and
- uses a learning factor α_t, which decreases gradually over time.

Q is updated as follows:

$$Q_{t+1}(S_t, A_t) \quad = \quad Q_t(S_t, A_t) + \alpha_t[r_t + \gamma \max_b Q_t(S_{t+1}, b) - Q_t(S_t, A_t)]. \qquad (3.10)$$

In Equation 3.10, we have indexed the Q-matrix by time, to show the incremental update of the Q-values given the prevailing values and new reward information. Provided each state-action pair is visited infinitely often, [244] show that Q converges to Q^* for any exploration policy. Singh et al. provide guidance as to specific exploration policies for asymptotic convergence to optimal actions and asymptotic exploitation under the *Q-learning* algorithm, which we incorporate in our analysis [230].

3.4.1 Proof of convergence for infinite-horizon *Q-learning*

While Watkins developed the *Q-learning* algorithm in his PhD thesis [243], the detailed proof of convergence was published as a separate technical note with Dayan [244], making use of an artificial controlled Markov process called an *action replay process*. Melo published an independent, simpler proof of convergence for the finite state, finite action, infinite horizon *Q-learning* algorithm which is well-suited to our formulation [182], building on theorems derived in [139]. We will reproduce Melo's proof here for completeness, making appropriate notation changes to suit our exposition, before considering the consequences of a finite horizon on convergence.

Consider an MDP as the tuple $(\mathcal{S}, \mathcal{A}, P, r)$, where

- $\mathcal{S}$ is the finite state space
- $\mathcal{A}$ is the finite action space, or set of admissible controls
- P represents the state transition probabilities
- r represents the reward function.

We denote elements of $\mathcal{S}$ as i and j, and elements of $\mathcal{A}$ as a and b. We consider the general situation where the reward is defined over triplets (i, a, j), viz. r is a function

$$r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$$

where a reward $r(i, a, j)$ is incurred each time a transition occurs from state i to state j, as a result of performing action a. Here, r is a bounded, deterministic function.

For a candidate sequence of controls $\mathcal{A}_u \subset \mathcal{A} = \{A_{u,t} = a : t \geq 0, a \in \mathcal{A}\}$ and state evolutions $\mathcal{S}_u \subset \mathcal{S} = \{S_{u,t} = i : t \geq 0, i \in \mathcal{S}\}$, the value of a state i is defined as

$$J(i, \mathcal{A}_u) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_{u,t}, A_{u,t}, S_{u,t+1}) \;\middle|\; S_{u,0} = i\right] \tag{3.11}$$

where γ is some discount rate such that $0 \leq \gamma \leq 1$. For each $i \in \mathcal{S}$, the optimal value function is defined as

$$V^*(i) = \max_{\mathcal{A}_u} J(i, \mathcal{A}_u) \tag{3.12}$$

which verifies the following equation,

$$V^*(i) = \max_{a \in \mathcal{A}} \sum_{j \in \mathcal{S}} P_a(i, j)\big[r(i, a, j) + \gamma V^*(j)\big]. \tag{3.13}$$

In line with the description in Section 3.3, we define the optimal Q-function as

$$Q^*(i,a) = \sum_{j \in \mathcal{S}} P_a(i,j)\big[r(i,a,j) + \gamma V^*(j)\big]. \tag{3.14}$$

The optimal Q-function in Equation 3.14 is a fixed point of a contraction operator $\mathbf{H}$, defined for some generic function $q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as

$$(\mathbf{H}q)(i,a) = \sum_{j \in \mathcal{S}} P_a(i,j)\big[r(i,a,j) + \gamma \max_{b \in \mathcal{A}} q(j,b)\big]. \tag{3.15}$$

The operator $\mathbf{H}$ is a contraction in the sup-norm, i.e.

$$||\mathbf{H}q_1 - \mathbf{H}q_2||_\infty \leq \gamma ||q_1 - q_2||_\infty. \tag{3.16}$$

To see this, consider the following

$$\begin{aligned}
&||\mathbf{H}q_1 - \mathbf{H}q_2||_\infty \\
&= \max_{i,a} \left| \sum_{j \in \mathcal{S}} P_a(i,j)\Big[r(i,a,j) + \gamma \max_{b \in \mathcal{A}} q_1(j,b) - r(i,a,j) - \gamma \max_{b \in \mathcal{A}} q_2(j,b)\Big] \right| \\
&= \max_{i,a} \gamma \left| \sum_{j \in \mathcal{S}} P_a(i,j)\Big[\max_{b \in \mathcal{A}} q_1(j,b) - \max_{b \in \mathcal{A}} q_2(j,b)\Big] \right| \\
&\leq \max_{i,a} \gamma \sum_{j \in \mathcal{S}} P_a(i,j) \Big| \max_{b \in \mathcal{A}} q_1(j,b) - \max_{b \in \mathcal{A}} q_2(j,b) \Big| \\
&\leq \max_{i,a} \gamma \sum_{j \in \mathcal{S}} P_a(i,j) \max_{k,b} \Big| q_1(k,b) - q_2(k,b) \Big| \\
&= \max_{i,a} \gamma \sum_{j \in \mathcal{S}} P_a(i,j) ||q_1 - q_2||_\infty \\
&= \gamma ||q_1 - q_2||_\infty.
\end{aligned}$$

The *Q-learning* algorithm determines the optimal Q-function using point samples. Let π_u be some random state-action policy mapping, such that

$$\mathbb{P}_{\pi_u}\big[A_{u,t} = a | S_{u,t} = i\big] > 0 \tag{3.17}$$

for all state-action pairs (i,a). Consider a sequence of actions $\mathcal{A}_u$, state evolutions $\mathcal{S}_u$ and rewards $R_u = \{r_t(i,a,j) : t \geq 0, i,j \in \mathcal{S}_u, a \in \mathcal{A}_u\}$. Then, for any initial state Q_0, *Q-learning* uses the following update rule

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t(S_t, A_t)\big[r_t + \gamma \max_{b \in \mathcal{A}} Q_t(S_{t+1}, b) - Q_t(S_t, A_t)\big] \tag{3.18}$$

where the step-sizes $\alpha_t(S_t, A_t)$ satisfy the condition $0 \leq \alpha_t(S_t, A_t) \leq 1$ and $r_t :=$

$r_t(S_t, A_t, S_{t+1})$. Q_t then represents the expected discounted reward of performing action A_t in state S_t, and then following the (current) optimal policy thereafter.

Before we can prove that the update rule in Equation 3.18 converges to the optimal Q-function, we require an auxiliary result from stochastic approximation theory.

Theorem 3.1. *The random iterative process* $\{\triangle_t\}$*, taking values in* $\mathbb{R}^n$ *and defined as*

$$\triangle_{t+1}(x) = (1 - \alpha_t(x))\triangle_t(x) + \alpha_t(x)F_t(x)$$

converges to zero w.p. 1 under the following assumptions:

- $0 \leq \alpha_t(x) \leq 1$, $\sum_t \alpha_t(x) = \infty$ *and* $\sum_t \alpha_t^2(x) < \infty$
- $||\mathbb{E}\big[F_t(x)|\mathcal{F}_t\big]||_W \leq \gamma||\triangle_t||_W$*, with* $0 \leq \gamma < 1$
- $\mathbb{V}\big[F_t(x)|\mathcal{F}_t\big] \leq C(1 + ||\triangle_t||_W^2)$*, for* $C > 0$.

where:

- $x \in \mathcal{S}$ *is a finite set*
- $\alpha_t(x)$ *is the time-dependent learning rate*
- $F_t(x)$ *is a time-dependent function of the state variables*
- $\mathcal{F}_t = \{F_t(x), F_{t-1}(x), ..., \alpha_t(x), \alpha_{t-1}(x), ..., \triangle_t(x), \triangle_{t-1}(x), ...\}$ *is the history up to time* t
- C, γ *are constants*
- $||\cdot||_W = \max_x |\frac{\cdot}{W(x)}|$ *is a weighted maximum norm*

Proof. The detailed proofs of this result and supporting lemmas are provided by Jaakkola et al. [139]. □

We can now state the following result:

Theorem 3.2. *Given a finite MDP* $(\mathcal{S}, \mathcal{A}, P, r)$*, the Q-learning algorithm given by the update rule*

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t(S_t, A_t)\big[r_t + \gamma \max_{b \in \mathcal{A}} Q_t(S_{t+1}, b) - Q_t(S_t, A_t)\big]$$

converges w.p. 1 to the optimal Q-function, provided

$$\sum_t \alpha_t(i,a) = \infty$$

and

$$\sum_t \alpha_t^2(i,a) < \infty$$

for all $(x,a) \in \mathcal{S} \times \mathcal{A}$*, i.e. all permissible state-action pairs.*

The conditions for $\alpha_t(i,a)$ in Theorem 3.2, together with the *Q-learning* update rule condition $0 \leq \alpha_t(i,a) \leq 1$, imply that all state-action pairs need to be visited infinitely often to ensure convergence to the optimal Q-function.

Proof. We begin by rewriting Equation 3.18 as

$$Q_{t+1}(S_t,A_t) = (1-\alpha_t(S_t,A_t))Q_t(S_t,A_t) + \alpha_t(S_t,A_t)\big[r_t + \gamma \max_{b\in\mathcal{A}} Q_t(S_{t+1},b)\big].$$

Subtracting the quantity $Q^*(S_t,A_t)$ from both sides and letting

$$\triangle_t(S_t,A_t) = Q_t(S_t,A_t) - Q^*(S_t,A_t)$$

yields

$$\triangle_{t+1}(S_t,A_t) = (1-\alpha_t(S_t,A_t))\triangle_t(S_t,A_t) + \alpha_t(S_t,A_t)\big[r_t + \gamma \max_{b\in\mathcal{A}} Q_t(S_{t+1},b) - Q^*(S_t,A_t)\big].$$

Now consider,

$$F_t(i,a) = r_t(i,a,X(i,a)) + \gamma \max_{b\in\mathcal{A}} Q_t(j,b) - Q^*(i,a),$$

where $X(i,a)$ is a random sample state taken from the Markov chain $(\mathcal{S}, P_a)$. Then we have

$$\begin{aligned}\mathbb{E}[F_t(i,a)|\mathcal{F}_t] &= \sum_{j\in\mathcal{S}} P_a(i,j)\Big[r_t(i,a,j) + \gamma \max_{b\in\mathcal{A}} Q_t(j,b) - Q^*(i,a)\Big] \\ &= (\mathbf{H}Q_t)(i,a) - Q^*(i,a).\end{aligned}$$

Using the fact that $Q^* = \mathbf{H}Q^*$, we have

$$\mathbb{E}[F_t(i,a)|\mathcal{F}_t] = (\mathbf{H}Q_t)(i,a) - (\mathbf{H}Q^*)(i,a).$$

It then follows from Equation 3.16 that

$$\begin{aligned} ||\mathbb{E}[F_t(i,a)|\mathcal{F}_t]||_\infty &\leq ||Q_t - Q^*||_\infty \\ &= \gamma||\triangle_t||_\infty. \end{aligned}$$

Finally, we have

$$\begin{aligned} &\mathbb{V}[F_t(x,a)|\mathcal{F}_t] \\ &= \mathbb{E}\left[\left(r_t(i,a,X(i,a)) + \gamma \max_{b\in\mathcal{A}} Q_t(j,b) - Q^*(i,a) - (\mathbf{H}Q_t)(i,a) + Q^*(i,a)\right)^2\right] \\ &= \mathbb{E}\left[\left(r_t(i,a,X(i,a)) + \gamma \max_{b\in\mathcal{A}} Q_t(j,b) - (\mathbf{H}Q_t)(i,a)\right)^2\right] \\ &= \mathbb{V}\left[r_t(i,a,X(i,a)) + \gamma \max_{b\in\mathcal{A}} Q_t(j,b)|\mathcal{F}_t\right] \end{aligned}$$

which, since r is bounded, verifies

$$\mathbb{V}\big[F_t(i,a)|\mathcal{F}_t\big] \leq C(1 + ||\triangle_t||_W^2)$$

for some constant C.

Then, by Theorem 3.1, $\triangle_t$ converges to zero w.p. 1, which implies Q_t converges to Q^* w.p. 1. □

3.4.2 On convergence for finite-horizon *Q-learning*

The exposition in Section 3.4.1 presents an algorithm which guarantees optimal policy convergence of a stationary infinite-horizon MDP. The stationarity assumption, and hence validity of the above result, needs to be questioned when considering a finite-horizon MDP, since states, actions and policies are time-dependent [213]. In particular, we are considering a discrete period, finite trading horizon, which guarantees execution of a given volume of shares. At each decision step in the trading horizon, it is possible to have different state spaces, actions, transition probabilities and reward values. Hence the above model needs revision. Garcia and Ndiaye consider this problem and provide a model specification which suits this purpose [96]. They propose a slight modification to the Bellman optimality equations shown above:

$$V_t^*(i) = \max_{a_t}\{r_t(i,a_t) + \gamma \sum_j P_{ij}^t(a_{t+1}) V_{t+1}^{\pi^*}(j)\}$$

for all $i \in S_t \subseteq \mathcal{S}, j \in S_{t+1} \subseteq \mathcal{S}$, $a_t \in A_t \subseteq \mathcal{A}$, $t \in \{0, 1, ..., T\}$ and $V^*_{T+1}(i) = 0$. This optimality equation has a single solution $V^* = \{V^*_1, V^*_2, ..., V^*_T\}$ that can be obtained using dynamic programming techniques. The equivalent discounted expected reward specification thus becomes:

$$Q^\pi_t(i, a_t) = r_t(i, a_t) + \gamma \sum_j P^t_{ij}(\pi(i)) V^\pi_{t+1}(j).$$

They propose a novel transformation of an T-step non-stationary MDP into an infinite-horizon process ([96]). This is achieved by adding an artifical final reward-free absorbing state x_{abs}, such that all actions $a_T \in \mathcal{A}$ taken at time T lead to x_{abs} with probability 1. Hence the revised *Q-learning* update equation becomes:

$$Q_{t+1}(i, a_t) = Q_t(i, a_t) + \alpha_t(i, a_t) U_t,$$

where

$$U_t = \begin{cases} r_t + \gamma \max_b Q_t(j, b) - Q_t(i, a_t) & \text{for } i \in S_t, j \in S_{t+1}, t < T \\ r_t - Q_t(i, a_t) & \text{for } i \in S_T \\ 0 & \text{otherwise.} \end{cases}$$

The learning rule for S_T is thus equivalent to setting $V^*_{T+1}(x_{abs}) = Q^*_{T+1}(x_{abs}, b) = 0 \; \forall$ $b \in A_{T+1} \subseteq \mathcal{A}$.

Garcia and Ndiaye further show that the above specification (in the case where $\gamma = 1$) will converge to the optimal policy with probability one, provided that each state-action pair is visited infinitely often, $\sum_t \alpha_t(i, a) = \infty$ and $\sum_t \alpha^2_t(i, a) < \infty$ [96].

3.5 Batch learning vs online learning

The traditional RL paradigm permits *online learning*, viz. an agent is deployed in an unknown domain, immediately begins taking actions according to some (initially random) action-selection policy, receives feedback from the system, and eventually refines its state-action selection policy through experience. *Batch RL* refers to the case where a certain amount of pre-recorded experience (rewards and state transitions given executed actions) is used to train a *Q-matrix* offline, before deploying the learning agent into the online environment [80, 158]. Batch RL does not preclude online learning, but rather seeks to minimise the potential adverse consequences of initial random action selection by seeding the *Q-matrix* with reasonable experience. It is of course only effective if the experience tuples for offline policy training closely mimic online experience [80]. In Chapter 5, we introduce a batch RL Q-learning framework for optimal trade execution,

where executed actions affect only the evolution of the agent's *private* state attributes, not the evolution of *public* attributes. This permits us to construct a set of experience tuples from historical data which can be used to seed the training of a *Q-matrix*, before deploying it online for trade execution decisions. Given the goal of developing adaptive trading agents, it is critical that the agent can extract and make sense of persistent information from streaming data *online*, and is able to converge to a *useful* policy fast enough before the regime changes. Chapter 6 introduces a market attribute which describes the temporal evolution of the system, but is measurable *online* for effective learning. Chapter 9 discusses a fully online algorithm, where the agent learns directly from interactions from the system with no offline training.

3.6 Exploration vs exploitation trade-off

A key challenge in RL, which is not a concern in other forms of machine learning, is the trade-off between *exploration* and *exploitation* [230]. Given the goal of maximising cumulative reward, an agent should choose actions which it has found to be successful in the past, i.e. *exploit* existing knowledge of the system to make decisions. At the same time, given limited knowledge of the system, the agent should *explore* actions it has not tried before to determine if it has perhaps learnt a locally-optimal policy. This trade-off is usually encoded in the action selection rule for the learning agent. A common action selection rule is ϵ-*greedy*, where most of the time the action chosen coincides with the highest expected cumulative future reward, however there is a ϵ probability of choosing a random action from the permissible set [230]. One disadvantage of ϵ-*greedy* is that, given an *exploration* action choice, it is equally likely to choose amongst all permissible actions. An alternative scheme is *softmax*, where action probabilities are weighted according to their estimated value [230], such that the *exploration* action choice does not adversely affect the agent's trajectory. We have chosen to use the ϵ-*greedy* action-selection scheme in Chapters 5, 8 and 9, however alternative schemes could be explored in further work to assess its efficacy in this domain.

3.7 Curse of dimensionality

The term *curse of dimensionality* was introduced by Richard Bellman to describe a problem of exponential increase the volume of a sampling space as one increases the dimension of the space [31–33], however the concept has been translated to machine learning, where it states that the efficacy of a learning algorithm decreases as the dimensionality of the feature space increases [135]. For RL, this is a particular problem

for the discrete-state *Q-learning* algorithm we consider in this thesis, as the ability to learn an effective policy quickly diminishes as the number of state attributes (and their resolution) increases. It is thus critical that we choose a parsimonious state representation which captures enough exploitable information in the system evolution to learn an effective policy. This is especially relevant for our domain, where agents acting in a high-frequency market microstructure state space potentially face solving high-dimensional problems. In Chapter 6, we introduce a scheme which models temporal evolution by efficiently extracting the exploitable information from multi-featured data streaming from the system, reducing it to low-dimensional representation which is easily measurable online. This permits us to add a single public attribute to our learning agent's state space, rather than adding all possible features at arbitrary resolutions.

3.8 The nature of learning in a complex system

While RL offers a compelling paradigm for deriving adaptive trajectories under Markovian dynamics, it rests firmly on the foundations of dynamic programming [29], assuming the system of interest is stationary and ergodic and hence guarantees convergence to a *single* globally-optimal state-action policy. If the system of interest is a complex adaptive system, the existence of a single, fixed-point solution for our objective should be questioned.

Rather, RL may still be effective if it converges to a *useful* policy *fast enough*, i.e. it learns at a rate faster than the natural time-scale of the system, yet adapts as new niches form. Galla and Farmer investigate the nature of agent learning by simulating two-person complicated games with varying payoff correlations, using experience weighted attraction (EWA) to evaluate the propensity for asymptotic learning in the parameter space [95]. They show that even under these simple conditions, chaotic regimes exist in the payoff correlation-learning rate space, where any attempts to learn a useful policy are inherently random. Their results suggest that, given a payoff correlation amongst competing agents, a learning rate should be chosen which ensures the agent is in a *fixed point* or *multiple fixed point* regime, such that feasible learning toward a useful policy is possible. Given the complex system paradigm we are considering, we would at best expect a (possibly dynamic) *multiple fixed point* regime to exist, thus a learning rate should be chosen which ensures we are within this region. It is unclear how the results presented by Galla and Farmer [95] could be extrapolated for real-world games. One possibility may be to use an inverse reinforcement learning (IRL) algorithm to learn rewards or payoffs of different classes of agents in financial systems, calculating payoff correlations to isolate feasible learning rates, but we suspect the solution is more

complicated. It is clear, however, that the chosen learning rate is critical for an effective trading agent and warrants careful evaluation for our study.

3.9 Some remarks

This chapter introduces the general learning framework used in this thesis: discrete-state, discrete-action *Q-learning* for finite-horizon MDPs. In Chapter 5, we demonstrate a simple implementation for our particular problem of optimal trade execution. Chapter 6 uses the complex system proposition for financial system dynamics outlined in Chapter 1 to inform a mechanism for detecting temporal states online, providing an efficient attribute to enhance the learning agent's state space which avoids the *curse of dimensionality*. Chapter 7 focuses on the agent's permitted actions, using the ideas outlined in Chapter 2 to inform a pragmatic mechanism for controlling the size of submitted trades.

Chapter 4

Data description and Exploratory Data Analysis

4.1 Overview

In this chapter, we discuss the raw data used for this work, courtesy of Thomson Reuters Tick History. To facilitate efficient storage, retrieval and manipulation of the large volume of tick data, we uploaded the data into a MongoDB noSQL database with bespoke query indexes and a MATLAB application programming interface (API) for integration into our scientific computing environment. We use the techniques of exploratory data analysis to interrogate the empirical data, revealing pertinent details for the modelling decisions which follow.

4.2 Data

4.2.1 Raw data

Thomson Reuters Tick History (TRTH) provides a historical market data service for intraday transactions, quotes and market depth dating back to January 1996 [2]. We used this service to collect raw tick data for 170 stocks which make up the South African benchmark All-Share index (ALSI), from January 2006 to December 2015. The TRTH web interface was used to select fields including transactions and 10 levels of market depth quotes, and data was downloaded as a series of flat text files. The fields retrieved are listed in Table 4.1.

Reuters field	**Description**
#RIC	The Reuters Sharecode identifier
Date[L]	The date, in *"DD-MMM-YY"* format
Time[L]	The local exchange time, in *"hh:mm:ss.000"* format, associated with the event
Type	The type of data shown (*"quote"*, *"trade"* or *"auction"*)
Price	The price of a trade
Size	The volume of a trade
L1-BidPrice	The prevailing best bid quote price
L1-BidSize	The prevailing best bid quote volume
L1-AskPrice	The prevailing best ask quote price
L1-AskSize	The prevailing best ask quote volume
L2-BidPrice	The prevailing level 2 bid quote price
L2-BidSize	The prevailing level 2 bid quote volume
L2-AskPrice	The prevailing level 2 ask quote price
L2-AskSize	The prevailing level 2 ask quote volume
L3-BidPrice	The prevailing level 3 bid quote price
L3-BidSize	The prevailing level 3 bid quote volume
L3-AskPrice	The prevailing level 3 ask quote price
L3-AskSize	The prevailing level 3 ask quote volume
L4-BidPrice	The prevailing level 4 bid quote price
L4-BidSize	The prevailing level 4 bid quote volume
L4-AskPrice	The prevailing level 4 ask quote price
L4-AskSize	The prevailing level 4 ask quote volume
L5-BidPrice	The prevailing level 5 bid quote price
L5-BidSize	The prevailing level 5 bid quote volume
L5-AskPrice	The prevailing level 5 ask quote price
L5-AskSize	The prevailing level 5 ask quote volume
L6-BidPrice	The prevailing level 6 bid quote price
L6-BidSize	The prevailing level 6 bid quote volume
L6-AskPrice	The prevailing level 6 ask quote price
L6-AskSize	The prevailing level 6 ask quote volume
L7-BidPrice	The prevailing level 7 bid quote price
L7-BidSize	The prevailing level 7 bid quote volume
L7-AskPrice	The prevailing level 7 ask quote price
L7-AskSize	The prevailing level 7 ask quote volume
L8-BidPrice	The prevailing level 8 bid quote price
L8-BidSize	The prevailing level 8 bid quote volume
L8-AskPrice	The prevailing level 8 ask quote price
L8-AskSize	The prevailing level 8 ask quote volume
L9-BidPrice	The prevailing level 9 bid quote price
L9-BidSize	The prevailing level 9 bid quote volume
L9-AskPrice	The prevailing level 9 ask quote price
L9-AskSize	The prevailing level 9 ask quote volume
L10-BidPrice	The prevailing level 10 bid quote price
L10-BidSize	The prevailing level 10 bid quote volume
L10-AskPrice	The prevailing level 10 ask quote price
L10-AskSize	The prevailing level 10 ask quote volume

TABLE 4.1: Thomson Reuters Tick History (TRTH) raw data fields. The associated data was downloaded as flat text files from the TRTH web interface.

4.2.2 MongoDB noSQL database

To facilitate efficient data retrieval for empirical analysis on microstructure data, we used MongoDB to implement a trade and quote database. MongoDB is a highly scalable, high performance, open-source NoSQL database management system (DBMS) [54]. Considering the nature of the raw tick data, i.e. large volumes of trades and quotes with associated metadata, the JSON-like documents and dynamic schemas provided by MongoDB ensure efficient storage, without the relational overhead of SQL-style databases. MongoDB offers simple and compound index support on any set of attributes, MapReduce functionality for efficient aggregation queries and full integration with JAVA and C++ via appropriate API's and drivers [54].

The implemented MongoDB schema is shown in Figure 4.1. A *TickData* database was created to store the data in two *collections*: *JSETransactions*, which stores trade and level-1 quotes as *documents*, and *JSEMarketDepth*, which stores 10 levels of market depth quotes as *documents*. The hardware specifications of the database server are shown in Table 4.2. Security measures were implemented to ensure the correct access permissions were granted for our research group team members, using role-based access control.

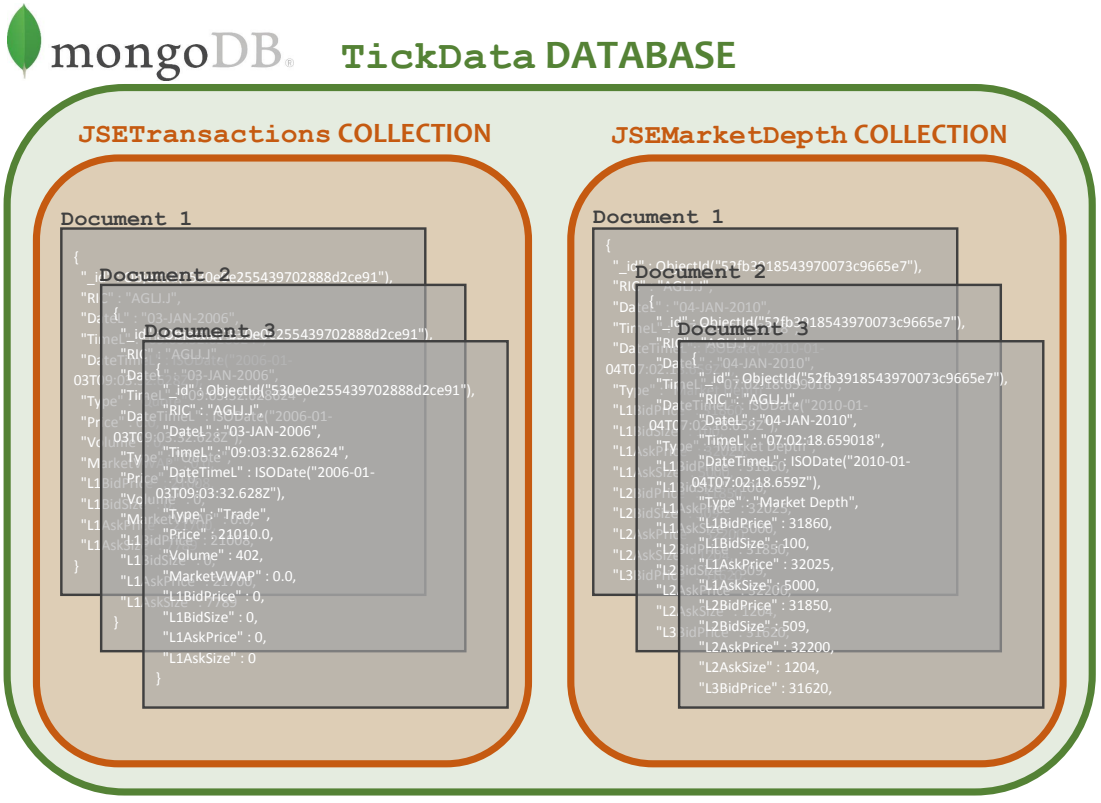


FIGURE 4.1: Implemented MongoDB schema for TRTH data. A *TickData* database was created to store the data in two *collections*: *JSETransactions*, which stores trade and level-1 quotes, and *JSEMarketDepth*, which stores 10 levels of market depth quotes.

Item	Specification
Operating System	Windows 7 Professional Service Pack 1 (64-bit)
DBMS	MongoDB 3.0.7, run as Windows service
CPU	Intel Core i7-X980 CPU@3.33 GHz
Memory	24GB RAM
HDD capacity (DB location)	3 x 4TB RAID 0 striped, creating spanned volume of 12TB
HDD capacity (raw data)	1 x 1.5TB
HDD capacity (OS)	1 x 1TB
GPU	Nvidia TESLA C2050 with 2.5GB RAM, CC: 2.0, SM: 2.0

TABLE 4.2: Hardware specifications for MongoDB data server.

4.2.2.1 Query indexes

MongoDB provides support for both *simple* (single field) and *compound* (multiple fields) indexes to improve the execution speed of queries. MongoDB make use of a B-tree data structure [60] which restricts the number of scans through a collection to those which are relevant for a given query. We constructed a number of compound indexes which facilitated efficient retrieval of documents from the *JSETransactions* and *JSEMarket-Depth* collections. Some of these are indicated in Figure 4.2. We typically required the retrieval of contiguous blocks of tick data, for a specified stock and date/time range, thus the specification of required indexes was relatively straight forward.

JSETransactions COLLECTION

Retrieve all documents for a given RIC (stock), type ('trade', 'quote' or 'auction') and date range using the ISODate format.

```
db.JSETransactions.createIndex({
      RIC: 1,
      Type: 1,
      DateTimeL: 1
})
```

Retrieve all documents for a given RIC (stock), and date range using the ISODate format (retrieves all types).

```
db.JSETransactions.createIndex({
      RIC: 1,
      DateTimeL: 1
})
```

JSEMarketDepth COLLECTION

Retrieve all documents for a given RIC (stock), and date range using the ISODate format (Collection only contains 'quote' types).

```
db.JSEMarketDepth.createIndex({
      RIC: 1,
      DateTimeL: 1
})
```

FIGURE 4.2: Some compound indexes created for efficient query execution.

4.2.2.2 Aggregation and Map-Reduce

MongoDB provides two schemes for native document processing and aggregation: The *aggregation pipeline* and *map-reduce*. The data *aggregation pipeline* allows the user to group and sort documents based on specified fields, as well as apply common operators on intermediate and final results for efficient computation using native MongoDB commands. The basic process uses queries to *filter* documents and then apply *document transformations* to modify the form of the output document. *Map-reduce* is an older paradigm implemented on a variety of large-scale computing clusters and database management systems [68]. Map-reduce operations typically have two phases: a *map* stage which processes each document and emits one or more objects for each input document, and *reduce* phase which combines and processes the output of the map operation. While the MongoDB *aggregation pipeline* is more efficient than *map-reduce* for performing the same operation, *map-reduce* offers more flexibility in the type of aggregation procedures which can be performed. Figure 4.3 illustrates the differences between the *aggregation pipeline* and *map-reduce* for performing a simple extraction of level-1 quotes at 5-minute intervals from the *JSEMarketDepth* collection.

4.2.2.3 MATLAB API

We developed a MATLAB application programming interface (API) which provides seamless access to data stored in the MongoDB database, efficiently retrieving documents over a secure connection and populating the MATLAB workspace for further computation. We incorporated the native *mongoexport* command, which efficiently extracts documents based on a specified query and stores results temporarily on disk, before populating the MATLAB workspace using the MATLAB *dos* command. Although this is a somewhat crude implementation, it was found to be significantly faster than alternative schemes which used Java Database Connectivity (JDBC) drivers to connect to the database from MATLAB.

AGGREGATION

Extract level 1 quotes at 5-minute intervals for GRT, from 25-Oct-2012 09:00 to 09:15

```
db.JSEMarketDepth.aggregate(
[ { $match:
      { RIC: "GRTJ.J", DateTimeL: { $gte: ISODate("2012-10-
      25T09:00:00.000Z"), $lte: ISODate("2012-10-
      25T09:15:00.000Z") } } },

      { $sort: { DateTimeL: 1 } },

      { $group: { _id: { RIC: "$RIC", Year: { $year:
      "$DateTimeL" }, Month: { $month: "$DateTimeL" }, Day:
      { $dayOfMonth: "$DateTimeL" }, Hour: { $hour:
      "$DateTimeL" }, Minute: { $minute: "$DateTimeL" } },

      L1AskPrice: { $first: "$L1AskPrice" }, L1AskSize: {
      $first: "$L1AskPrice" }, L1BidPrice: { $first:
      "$L1BidPrice" }, L1BidSize: { $first: "$L1BidSize" }
      } },

      { $sort: { "_id.Year": 1, "_id.Month": 1, "_id.Day":
      1, "_id.Hour": 1, "_id.Minute": 1 } },

      { $match: { "_id.Minute": { $mod: [5, 0] } } } ] )
```

MAP-REDUCE

Extract level 1 quotes at 5-minute intervals for GRT

```
db.JSEMarketDepth.mapreduce(
map() {     var resolution = 5;
            var coeff = 1000 * 60 * resolution;
            var roundTime = new
            Date(Math.round(this.DateTimeL.getTime() /
            coeff) * coeff + 7200000);

      emit( {     RIC: this.RIC, Date: new Date(roundTime),
                  Year: roundTime.getFullYear(),
                  Month: roundTime.getMonth(),
                  Day: roundTime.getUTCDate(),
                  Hour: roundTime.getUTCHours(),
                  Minute: roundTime.getMinutes()},
              {   count: 1,
                  L1BidPrice: this.L1BidPrice,
                  L1BidSize: this.L1BidSize,
                  L1AskPrice: this.L1AskPrice,
                  L1AskSize: this.L1AskSize
              } );
      }
reduce(key, values) {
      var reduced = { count: 0, L1BidPrice: 0, L1BidSize:
      0, L1AskPrice: 0, L1AskSize: 0 };

      for (var i = 0; i<values.length; i++)
      {
            reduced.L1BidPrice = values[i].L1BidPrice;
            reduced.L1BidSize = values[i].L1BidSize;
            reduced.L1AskPrice = values[i].L1AskPrice;
            reduced.L1AskSize = values[i].L1AskSize;
      }
      return reduced;
} )
```

FIGURE 4.3: Aggregation pipeline versus map-reduce for a simple computation, extracting level-1 quotes for a stock at 5-minute intervals.

4.3 Exploratory Data Analysis

Initially promoted by John Tukey, exploratory data analysis (EDA) is a paradigm which seeks to construct visualisations which expose patterns and features of the data and reveal these *forcefully* to the analyst [129]. The term *forcefully* distinguishes mere visual summaries of data from those which promote insight and investigation. Tukey claimed that statisticians were too focused on sometimes arbitrary statistical hypothesis testing or *confirmatory data analysis*, rather than efficiently utilising the data to inform which hypotheses indeed needed testing [234, 235]. In a key publication, *The Future of Data Analysis*, Tukey provides some key propositions in the interest of gleaning new and meaningful insights from datasets, advocating the use of more realistic frameworks for investigating old problems, seeking unfamiliar summaries of observational material and the consideration of the perspectives of different experts on the same data [233].

Given the proliferation of data being generated by a wide variety of domains, these principles of EDA should be reconceptualised to inform hypotheses in the so-called era

of *big data* [90]. Advances in techniques which were conceptualised for *small data* will require a careful blend of domain-specific knowledge and statistics to ensure inferences are both meaningful and significant. We will thus consider EDA specifically for financial markets, using a blend of our understanding of system dynamics and mechanics with observed behaviours to inform hypotheses about system evolution.

4.3.1 Visualisation of limit order book features

Figure 4.4 demonstrates the asynchronous nature of trade prices as they are recorded in the data feed. Each dot indicates a trade which occurred over the 5-minute time interval shown here, where dots are coloured by stock and sized by the quantity of the trade. This reveals the varying event throughput across stocks and apparent clustering of activity.

Figure 4.5 plots all trade events for all TOP40 stocks for a typical trading day. While certain details are masked at the micro level in this figure, a macro lens reveals that there are certain temporal behavioural patterns worth investigating, such as clustering of high-volume trades, their coincidence with exogenous events and news, as well as cross-sectional activity levels for multiple stocks at various stages in the trading day.

Figures 4.6 and 4.7 plot the *best bid* (red) and *best ask* (blue) quotes for two candidate stocks (SBK and AGL) over a typical trading day. Here, we note the variability of the *spread* over the trading day, and the effects of opening and closing auctions on top-of-book quote dynamics. This begins to suggest isolating continuous trading times (excluding auctions) for effective model calibrations, as inclusion of auction time dynamics may adversely affect calibrations. Figure 4.7 also suggests that certain exogenous events, such as the UK and US market open (around 10:00 and 15:00 respectively) may affect the dynamics of stocks on the local exchange.

Figure 4.8 plots the *best bid* and *best ask* quotes for two fundamentally similar stocks, Mondi Limited (MND) and Mondi Plc (MNP). These plots may reveal pairs trading behaviour amongst the submitted quotes and resultant trades, revealing details about how the observed price levels remain coupled.

Figures 4.9, 4.10 and 4.11 plot the market depth quotes and trades over a two hour period on a typical trading day. Here, blue dots indicate *ask quotes*, red dots indicate *bid quotes* and yellow dots indicate *trades*, with the size of the dot proportional to the associated quantity. Quote levels further from the top of the book are indicated in a lighter colour. These plots summarise a number of interesting features: clustering of trades, variability of spread, persistency of quotes before matching, cancellation or

updating and the tightness of market depth. Figure 4.11 in particular reveals a definite regime change in LOB activity for AGL following the UK market open at 10:00.

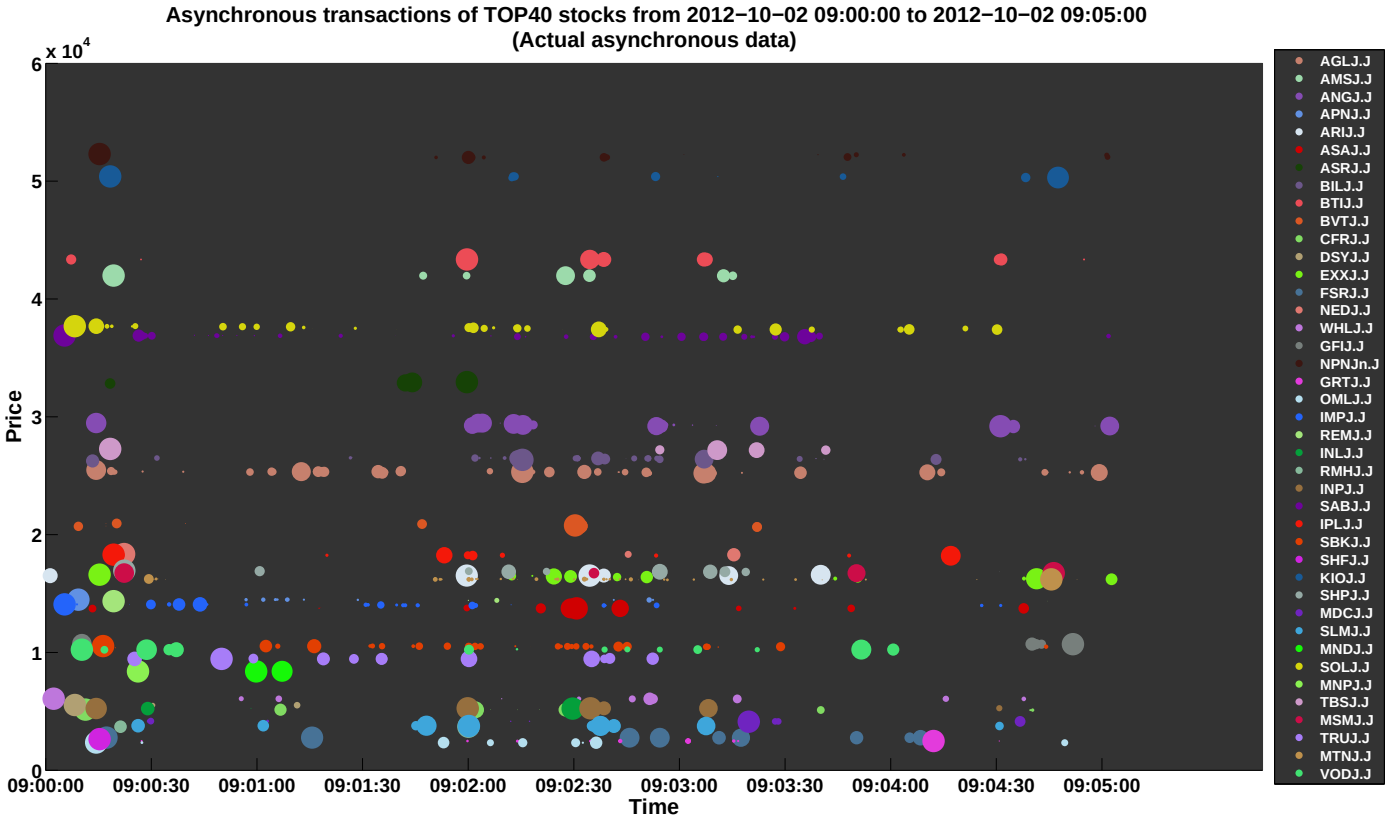


FIGURE 4.4: This figure aims to demonstrate the asynchronous nature of the trade price time series at the tick level. Here, we plot raw trades of all TOP40 stocks over a 5 minute period, from 09:00 02 October 2012 to 09:05 02 October 2012. Each dot represents a trade at the exact time it took place, with trade price on the Y-axis and time on the X-axis, where the size of the dot is proportional to the volume of the trade. Dots are coloured by stock. It is clear stock trades occur asynchronously.

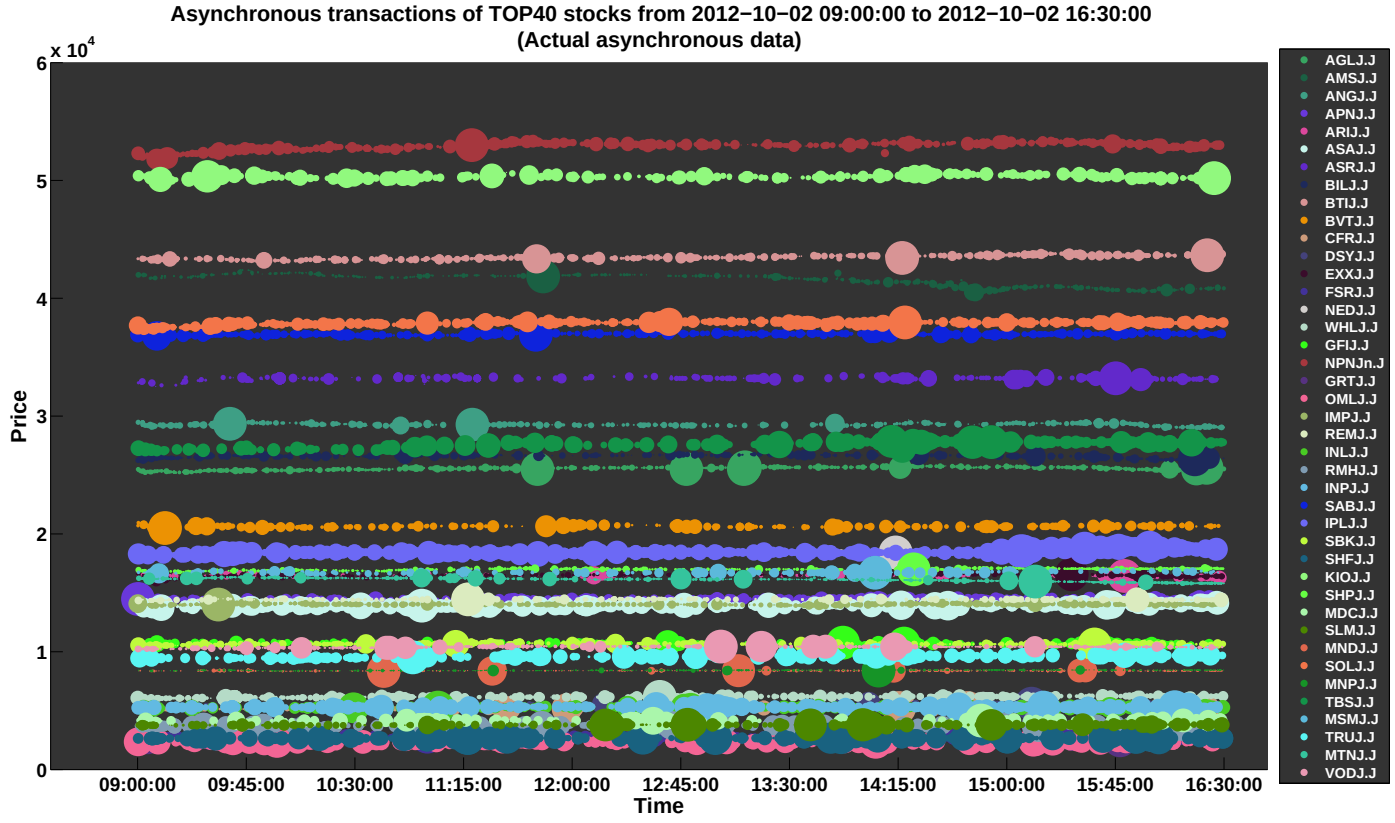


FIGURE 4.5: This figure aims to investigate common temporal patterns amongst stock trades across the trading day. Here, we plot raw trades of all TOP40 stocks over a 7.5 hour period, from 09:00 02 October 2012 to 16:30 02 October 2012. Each dot represents a trade at the exact time it took place, with trade price on the Y-axis and time on the X-axis, where the size of the dot is proportional to the volume of the trade. Dots are coloured by stock.

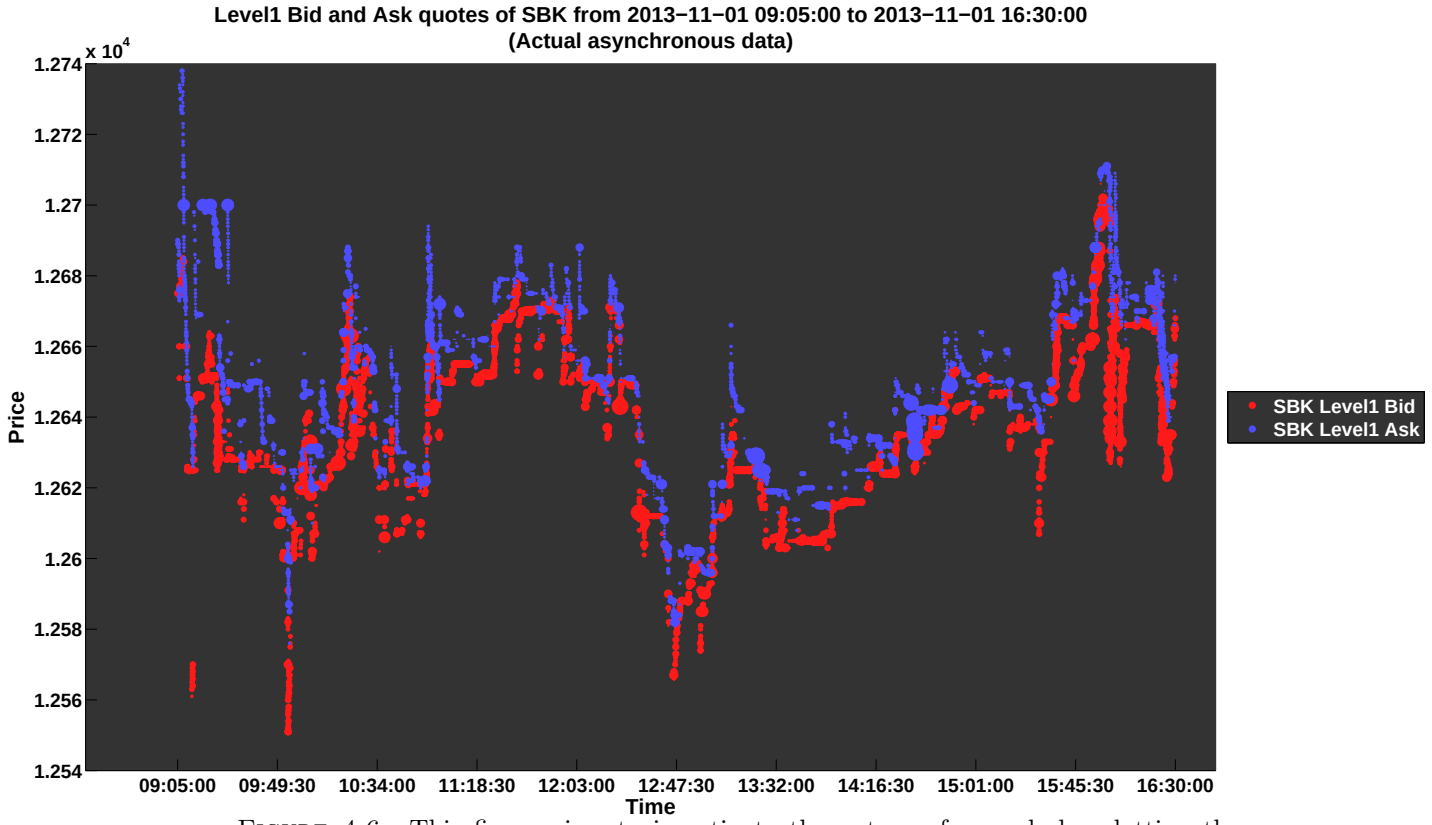


FIGURE 4.6: This figure aims to investigate the nature of spreads by plotting the evolution of level-1 quotes.

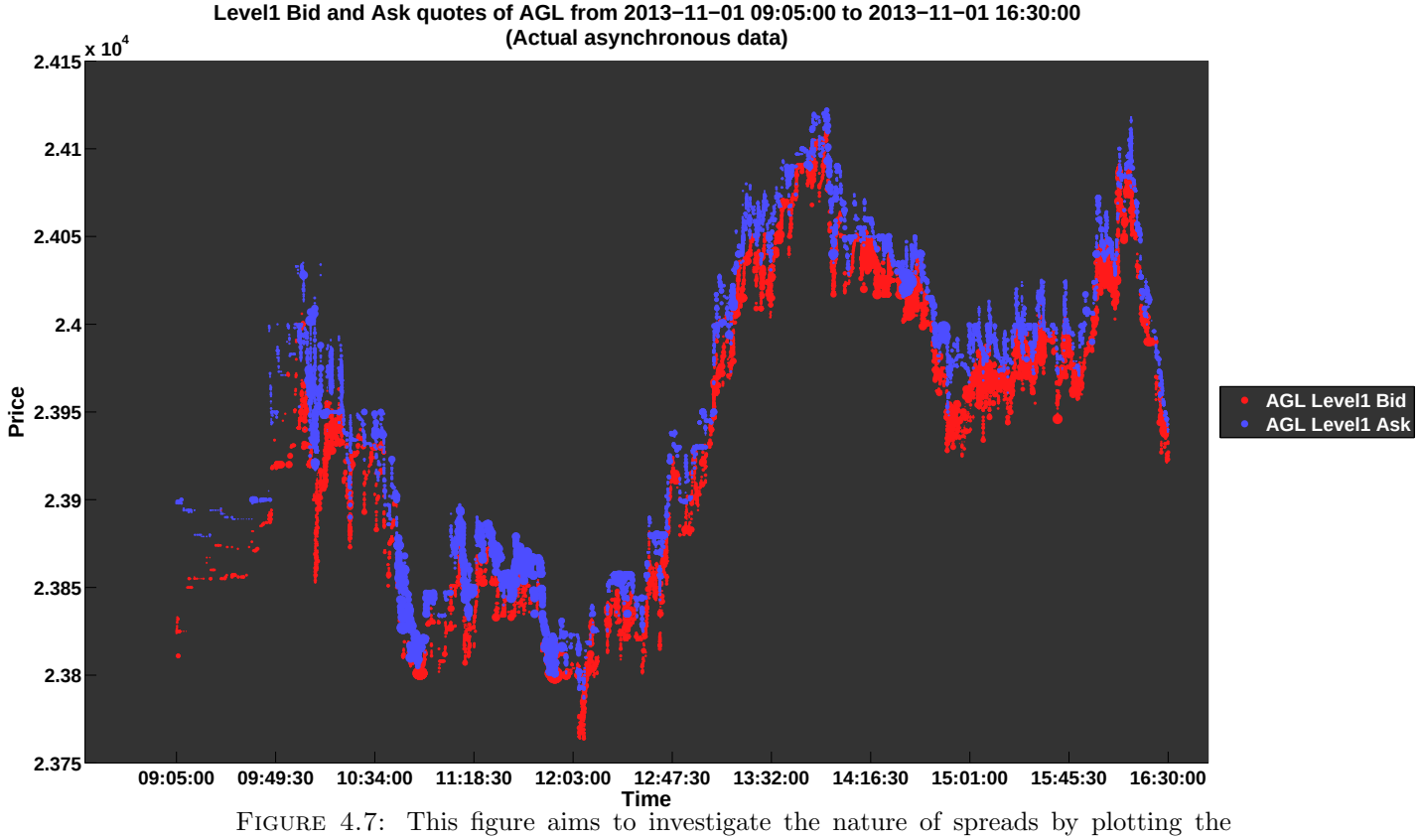


FIGURE 4.7: This figure aims to investigate the nature of spreads by plotting the evolution of level-1 quotes.

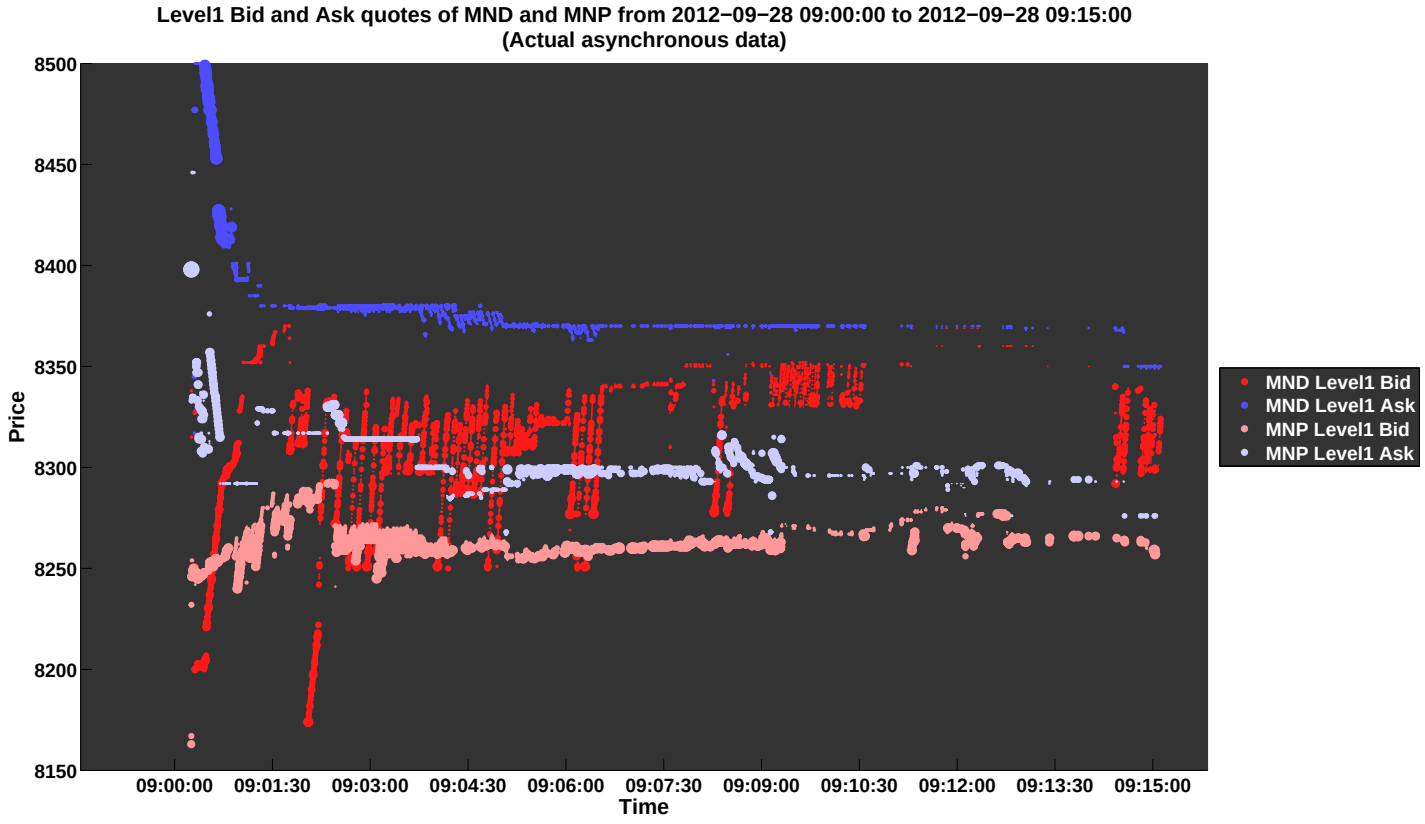


FIGURE 4.8: This figure aims to investigate the nature of level-1 quote updates for two fundamentally similar stocks which are typical candidates for pairs trading, Mondi Limited (MND) and Mondi Plc (MNP).

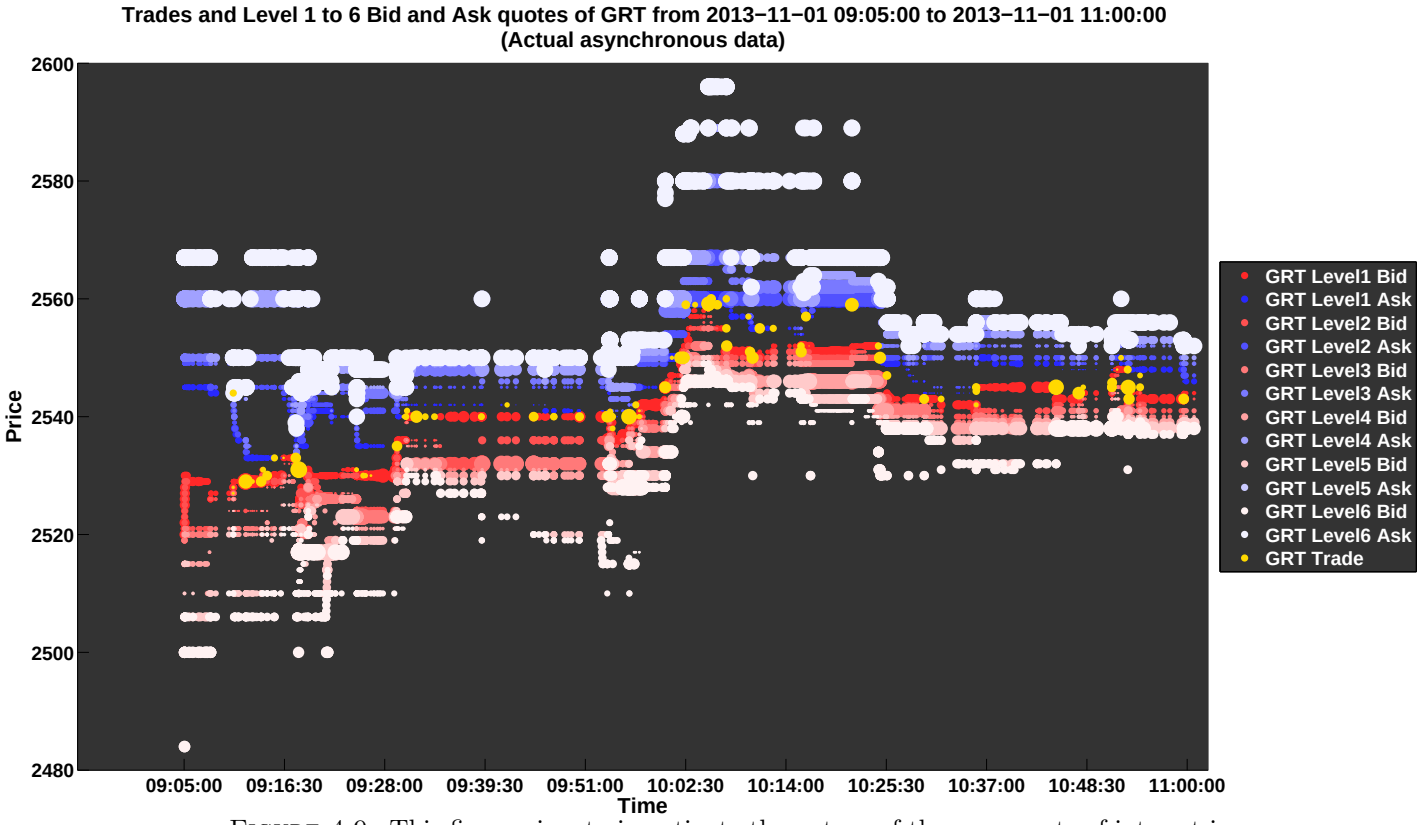


FIGURE 4.9: This figure aims to investigate the nature of the raw events of interest in the limit order book, including market depth. Ask quotes are indicated in blue and bid quotes in red, with darker colours indicating closeness to the top-of-the-book. Yellow dots indicate trade events. The size of each dot is proportional to the volume of the trade or quote.

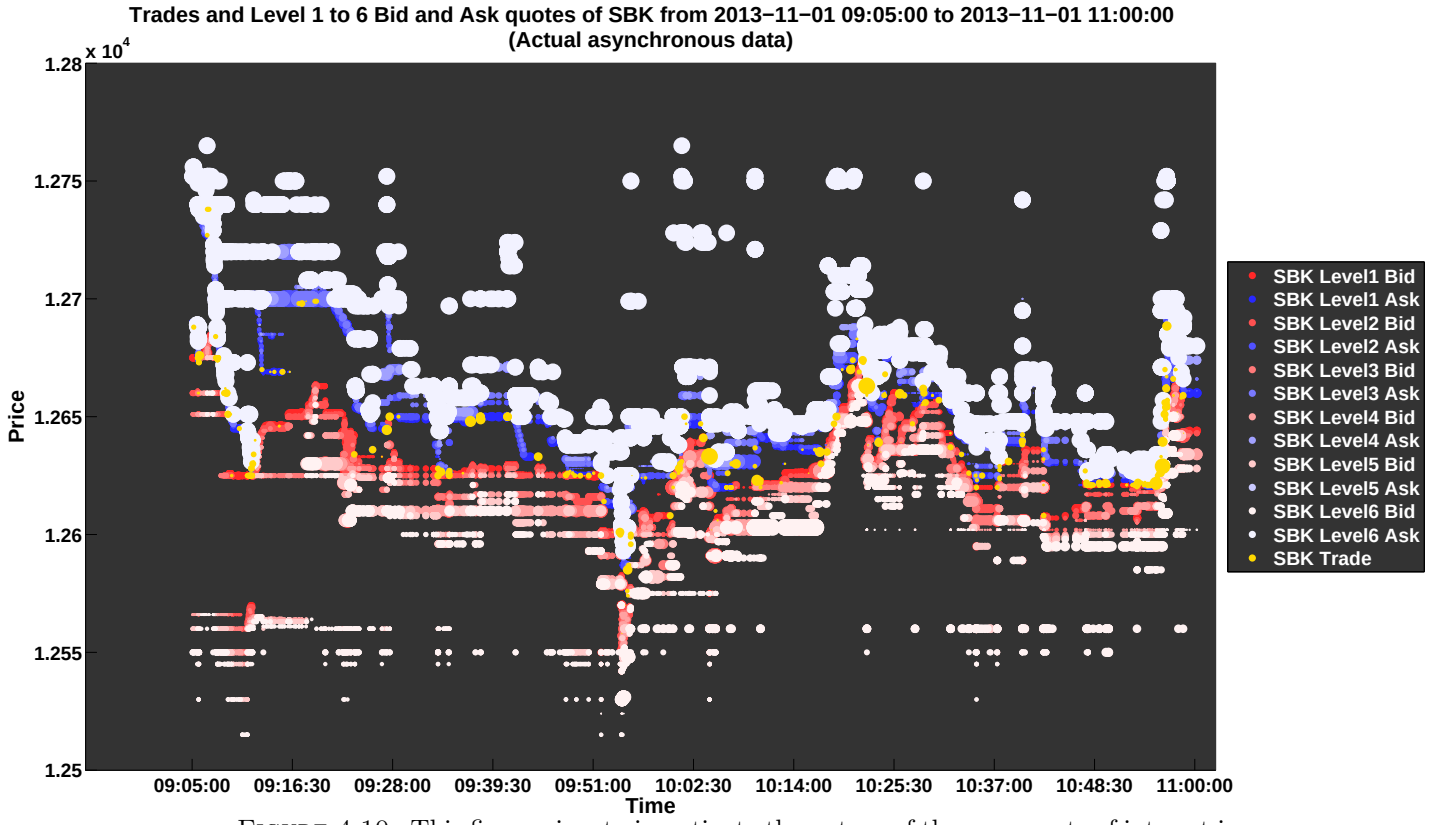


FIGURE 4.10: This figure aims to investigate the nature of the raw events of interest in the limit order book, including market depth. Ask quotes are indicated in blue and bid quotes in red, with darker colours indicating closeness to the top-of-the-book. Yellow dots indicate trade events. The size of each dot is proportional to the volume of the trade or quote.

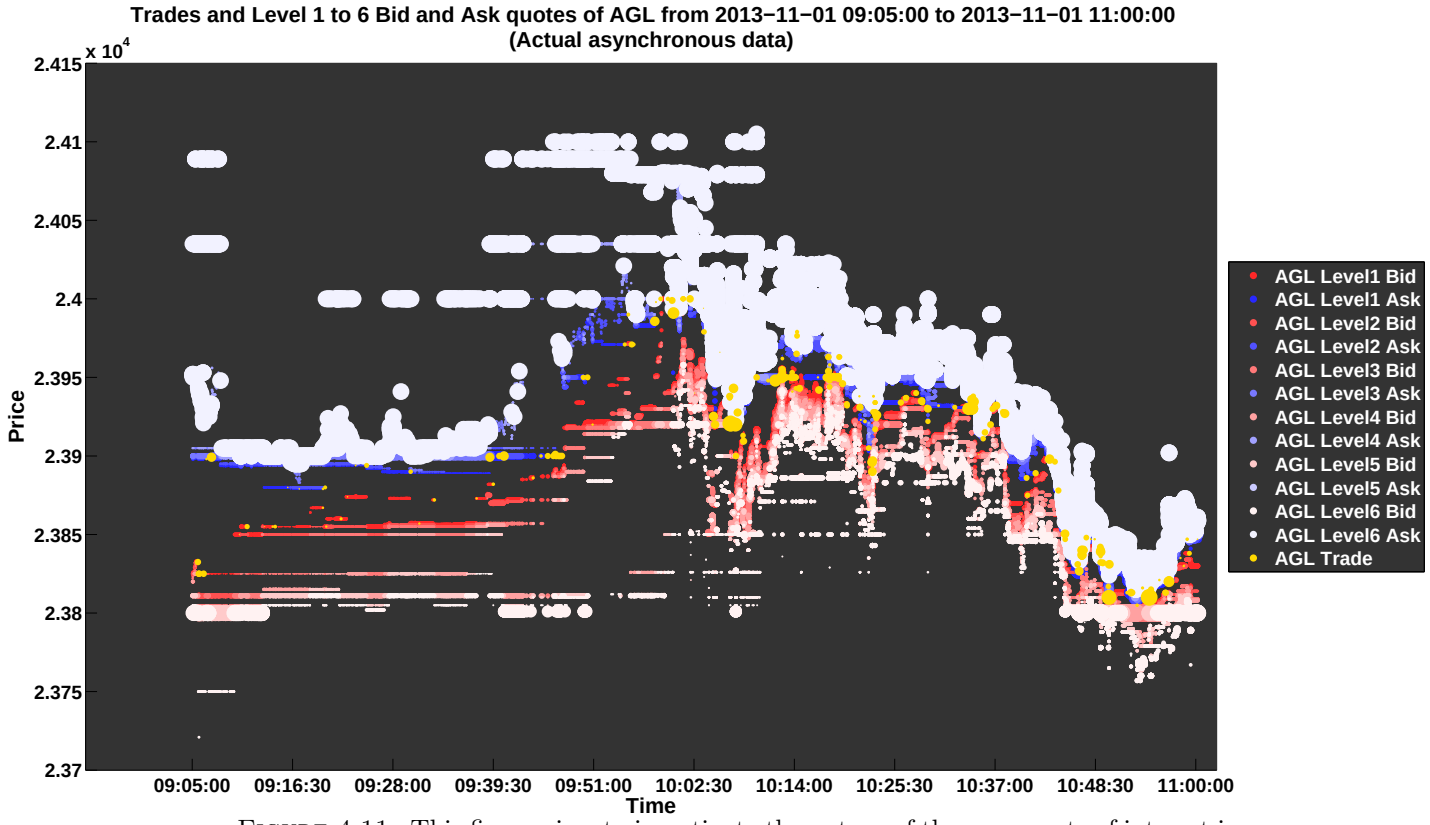


FIGURE 4.11: This figure aims to investigate the nature of the raw events of interest in the limit order book, including market depth. Ask quotes are indicated in blue and bid quotes in red, with darker colours indicating closeness to the top-of-the-book. Yellow dots indicate trade events. The size of each dot is proportional to the volume of the trade or quote.

4.4 Some remarks

This chapter highlights the importance of an effective database management system when dealing with large volumes of financial tick data, and uses the EDA paradigm of John Tukey to construct visualisations of the data which promote preliminary insights. While there are many more effective visualisations one could develop for this data, the investigation here reveals apparent asynchronicity, clustering and temporal behaviour of LOB events, which prove to be critical insights to inform the model choices which follow in this thesis.

Chapter 5

A simple model-free reinforcement learning model for trade execution

5.1 Overview

This chapter serves as a proof-of-concept for the proposed learning paradigm, distilling the problem to its most basic form before considering refinement. We introduce a model-free reinforcement learning algorithm for optimal trade execution, using pre-processed features to enumerate a discrete state space, with market order volume as the chosen control. The learning algorithm is used to adapt a static liquidation trajectory with respect to prevailing order book features, in order to improve the post-trade implementation shortfall with respect to the program's arrival price. The contribution is captured in the following paper:

D. Hendricks, D. Wilcox. *A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution.* Proceedings from IEEE Conference on Computational Intelligence for Financial Economics and Engineering, 2014. [124]
Available online: http://dx.doi.org/10.1109/CIFEr.2014.6924109

5.2 Adapting a static liquidation trajectory using reinforcement learning

A critical problem faced by participants in investment markets is the so-called *optimal trade execution* problem, viz. how *best* to trade a given block of shares to achieve *minimal cost*. Here, cost can be interpreted as in Andre Perold's implementation shortfall [209], i.e. adverse deviations of actual transaction prices from an arrival price baseline when the investment decision is made. Alternatively, cost can be measured as a deviation from the market volume-weighted trading price (VWAP) over the trading period, effectively comparing the specific trader's performance to that of the average market trader. In each case, the primary problem faced by the trader/execution algorithm is the compromise between price impact and opportunity cost when executing an order.

Price impact here refers to adverse price moves due to a large trade size absorbing liquidity supply at available levels in the order book (temporary price impact). As market participants begin to detect the total volume being traded, they may also adjust their bids/offers downward/upward to anticipate order matching (permanent price impact) [131]. To avoid price impact, traders may split a large order into smaller child orders over a longer period. However, there may be exogenous market forces which result in execution at adverse prices (opportunity cost). This behaviour of institutional investors was empirically demonstrated by Chan and Lakonishok [52], where they observed that typical trades of large investment management firms are almost always broken up into smaller trades and executed over the course of a day or several days.

Several authors have studied the problem of optimal liquidation, with a strong bias towards stochastic dynamic programming solutions (see [8–13, 35, 99, 100, 134, 160, 166, 190, 218–222, 236, 237] as examples). In this chapter, we consider the application of a machine learning technique to the problem of optimal liquidation. Specifically, we consider a case where the popular Almgren-Chriss closed-form solution for a trading trajectory, with linear price impact [13], can be enhanced by exploiting microstructure attributes over the trading horizon using a reinforcement learning technique.

Reinforcement learning in this context is essentially a calibrated policy mapping states to optimal actions. Each state is a vector of observable attributes which describe the current configuration of the system. It proposes a simple, model-free mechanism for agents to learn how to act optimally in a controlled Markovian domain, where the quality of action chosen is successively improved for a given state [243]. For the optimal liquidation problem, the algorithm examines the salient features of the current order book and current state of execution in order to decide which action (e.g. child order price or volume) to select to service the ultimate goal of minimising cost.

The first documented large-scale empirical application of reinforcement learning algorithms to the problem of optimised trade execution in modern financial markets was conducted by Nevmyvaka et al. [191, 192]. They set up their problem as a minimisation of implementation shortfall for a buying/selling program over a fixed time horizon with discrete time periods. For actions, the agent could choose a price to repost a limit order for the remaining shares in each discrete period. State attributes included elapsed time, remaining inventory, current spread, immediate cost and signed volume. In their results, they found that their reinforcement learning algorithm improved the execution efficiency by 50% or more over traditional submit-and-leave or market order policies. This is conceptually similar to work of Laruelle et al. [161], where they propose a stochastic optimisation procedure to determine the optimal posting of limit order prices given market feedback.

Given the above description, we are able to discuss our specific choices for state attributes, actions and rewards in the context of the optimal liquidation problem. We need to consider a specification which adequately accounts for our state of execution and the current state of the limit order book, representing the opportunity set for our ultimate goal of executing a volume of shares over a fixed trading horizon. We consider the particular problem of adapting a given, static volume trajectory for a liquidation program with respect to market micostructure features. Our candidate static trajectory model is the Almgren-Chriss model for an arrival price benchmark, assuming linear price impact.

5.2.1 The Almgren-Chriss model for optimal liquidation

Bertsimas and Lo are pioneers in the area of optimal liquidation, treating the problem as a stochastic dynamic programming problem [35]. They employed a dynamic optimisation procedure which finds an explicit closed-form best execution strategy, minimising trading costs over a fixed period of time for large transactions. Almgren and Chriss extended the work of Bertsimas and Lo to allow for risk aversion in their framework [13]. They argue that incorporating the uncertainty of execution of an optimal solution is consistent with a trader's utility function. In particular, they employ a price process which permits linear permanent and temporary price impact functions to construct an efficient frontier of optimal execution. They define a trading strategy as being *efficient* if there is no strategy which has lower execution cost variance for the same or lower level of expected execution cost.

The exposition of their solution is as follows: They assume that the security price evolves according to a discrete arithmetic random walk:

$$S_k = S_{k-1} + \sigma\tau^{1/2}\xi_k - \tau g(\frac{n_k}{\tau}), \tag{5.1}$$

where:

$$\begin{aligned}
S_k &= \text{ price at time } k,\\
\sigma &= \text{ volatility of the security},\\
\tau &= \text{ length of discete time interval},\\
\xi_k &= \text{ draws from independent random variables},\\
n_k &= \text{ volume traded at time } k \text{ and}\\
g(.) &= \text{ permanent price impact.}
\end{aligned}$$

Here, permanent price impact refers to changes in the equilibrium price as a direct function of our trading, which persists for at least the remainder of the liquidation horizon. Temporary price impact refers to adverse deviations as a result of absorbing available liquidity supply, but where the impact dissipates by the next trading period due to the resilience of the order book. Almgren and Chriss introduce a temporary price impact function $h(v)$ to their model, where $h(v)$ causes a temporary adverse move in the share price as a function of our trading rate v [13]. Given this addition, the actual security transaction price at time k is given by:

$$\tilde{S_k} = S_{k-1} - h(\frac{n_k}{\tau}).$$

Assuming a *sell* program for a quantity of X shares, we can then define the total trading revenue as:

$$\sum_{k=1}^{N} n_k\tilde{S_k} = XS_0 + \sum_{k=1}^{N}(\sigma\tau^{1/2}\xi_k - \tau g(\frac{n_k}{\tau}))x_k - \sum_{k=1}^{N} n_k h(\frac{n_k}{\tau}), \tag{5.2}$$

where $x_k = X - \sum_{j=1}^{k} n_j = \sum_{j=k+1}^{N} n_j$ for $k = 0, 1, ..., N$.

The total cost of trading is thus given by $x = XS_0 - \sum n_k\tilde{S_k}$, i.e. the difference between the target revenue value and the total actual revenue from the execution. This definition refers to Perold's implementation shortfall measure [209], and serves as the primary transaction cost metric which is minimised in order to maximise trading revenue. Since implementation shortfall is a random variable, Almgren and Chriss compute the

following:

$$\mathbb{E}(x) := \sum_{k=1}^{N} \tau x_k g(\frac{n_k}{\tau}) + \sum_{k=1}^{N} n_k h(\frac{n_k}{\tau})$$

and

$$\mathbb{V}(x) := \sigma^2 \sum_{k=1}^{N} \tau {x_k}^2.$$

The distribution of implementation shortfall is Gaussian if the ξ_k are Gaussian.

Given the overall goal of minimising execution costs and the variance of execution costs, they specify their objective function as:

$$\min_x \{\mathbb{E}(x) + \lambda \mathbb{V}(x)\}, \tag{5.3}$$

where:

$$\begin{aligned} x = &\text{ implementation shortfall,} \\ \lambda = &\text{ level of risk aversion.} \end{aligned}$$

The intuition of this objective function can be thought of as follows: Consider a stock which exhibits high price volatility and thus a high risk of price movement away from the reference price. A risk averse trader would prefer to trade a large portion of the volume immediately, causing a (known) price impact, rather than risk trading in small increments at successively adverse prices. Alternatively, if the price is expected to be stable over the liquidation horizon, the trader would rather split the trade into smaller sizes to avoid price impact. This trade-off between speed of execution and risk of price movement is what governs the shape of the resulting trade trajectory in the AC framework.

A detailed derivation of the general solution can be found in [13]. Here, we state the general solution:

$$x_j = \frac{\sinh(\kappa(T - t_j))}{\sinh(\kappa T)} X \text{ for } j = 0, ..., N. \tag{5.4}$$

The associated trade list is:

$$n_j = \frac{2\sinh(\frac{1}{2}\kappa\tau)}{\sinh(\kappa T)} \cosh(\kappa(T - t_{j-\frac{1}{2}}))X \text{ for } j = 0, ..., N, \tag{5.5}$$

where:

$$\kappa = \frac{1}{\tau}\cosh^{-1}\left(\frac{\tau^2}{2}\tilde{\kappa}^2 + 1\right),$$
$$\tilde{\kappa}^2 = \frac{\lambda\sigma^2}{\eta(1 - \frac{\rho\tau}{2\eta})},$$
$$\eta = \text{temporary price impact parameter},$$
$$\rho = \text{permanent price impact parameter},$$
$$\tau = \text{length of discrete time period}.$$

This implies that for a program of selling an initially long position, the solution decreases monotonically from its initial value to zero at a rate determined by the parameter κ. If trading intervals are short, κ^2 is essentially the ratio of the product of volatility and risk-intolerance to the temporary transaction cost parameter. We note here that a larger value of κ implies a *more rapid* trading program, again conceptually confirming the propositions of [134] that an intolerance for execution risk leads to a larger concentration of quantity traded early in the trading program. Another consequence of this analysis is that different sized baskets of the same securities will be liquidated in the same manner, barring scale differences and provided the risk aversion parameter λ is held constant. This may be counter-intuitive, since one would expect larger baskets to be effectively less liquid, and thus follow a *less rapid* trading program to minimise price impact costs.

It should be noted that the AC solution yields a suggested volume trajectory over the liquidation horizon, however Almgren and Chriss do not discuss a prescribed *order type* to execute the trade list [13]. We have assumed that the trade list can be executed as a series of *market orders*. Given that this implies we are always crossing the spread, one needs to consider that traversing an order book with thin volumes and widely-spaced prices could have a significant transaction cost impact. We thus consider a reinforcement learning technique which learns *when* and *how much* to cross the spread, based on the current order book dynamics.

The general solution outlined above assumes linear price impact functions, however the model was later extended by Almgren to account for non-linear price impact [12]. This extended model can be considered as an alternative base model in future research.

5.2.2 State space

We acknowledge that the true complexity of the financial system cannot be distilled into a finite set of states and is not likely to evolve according to a Markov process. However, we conjecture that the essential features of the system can be sufficiently captured with

some simplifying assumptions such that meaningful insights can still be inferred. Here we consider typical pre-processed features which capture aspects of the system visible to human traders. For simplicity, we have chosen a look-up table representation of Q, where Q is a 2-dimensional matrix where each state-action pair reflects the expected discounted future reward of performing the associated action in the given state, then following the (current) optimal policy thereafter. Function approximation variants may be explored in future research for more complex system configurations. As described above, each state $z_n \in \mathcal{S}$ represents a vector of observable attributes which describe the configuration of the system at time n. As in Nevmyvaka et al. [191, 192], we use *Elapsed Time* t and *Remaining Inventory* i as private attributes which capture our state of execution over a finite liquidation horizon T. Since our goal is to modify a given volume trajectory based on favourable market conditions, we include *spread* and *volume* as candidate market attributes. The intuition here is that the agent will learn to increase (decrease) trading activity when *spreads* are narrow (wide) and *volumes* are high (low). This would ensure that a more significant proportion of the total volume-to-trade would be secured at a favourable price and, similarly, less at an unfavourable price, ultimately reducing the post-trade implementation shortfall. Given the look-up table implementation, we have simplified each of the state attributes as follows:

- T = Trading Horizon,
- V = Total Volume-to-Trade,
- H = Hour of day when trading will begin,
- I = Number of remaining inventory states,
- B = Number of spread states,
- W = Number of volume states,
- sp_n = %ile Spread of the nth tuple,
- vp_n = %ile Bid/Ask Volume of the nth tuple,
- **Elapsed Time**: $t_n = 1, 2, 3, ..., T$,
- **Remaining Inventory**: $i_n = 1, 2, 3, ..., I$,
- **Spread State**: $s_n = \begin{cases} 1, & \text{if } 0 < sp_n \leq \frac{1}{B} \\ 2, & \text{if } \frac{1}{B} < sp_n \leq \frac{2}{B} \\ ... & \\ B, & \text{if } \frac{(B-1)}{B} < sp_n \leq 1, \end{cases}$

- **Volume State**: $v_n = \begin{cases} 1, & \text{if } 0 < vp_n \leq \frac{1}{W} \\ 2, & \text{if } \frac{1}{W} < vp_n \leq \frac{2}{W} \\ ... & \\ W, & \text{if } \frac{(W-1)}{W} < vp_n \leq 1. \end{cases}$

Thus, for the nth episode, the state attributes can be summarised as the following tuple:

$$z_n =< t_n, i_n, s_n, v_n > .$$

For sp_n and vp_n, we first construct a historical distribution of spreads and volumes based on the training set. It has been empirically observed that major equity markets exhibit U-shaped or J-shaped trading intensity curves throughout the day, i.e. varying signatures of typical trading activity around mornings, noon and afternoons. A further discussion of these insights can be found in Admati and Pfleiderer [6] and Brock and Kleidon [48]. In fact, Du Preez empirically demonstrates that South African stocks exhibit U-shaped volume and J-shaped spread characteristics over the trading day [74]. We thus consider simulations where training volume/spread tuples are H-hour dependent, such that the optimal policy is further refined with respect to trading time (H).

5.2.3 Action set

Based on the Almgren-Chriss (AC) model specified above, we calculate the AC volume trajectory $(AC_1, AC_2, ..., AC_T)$ for a given volume-to-trade (V), fixed time horizon (T) and discrete trading periods ($t = 1, 2, ..., T$). AC_t represents the proportion of V to trade in period t, such that $\sum_{t=1}^{T} AC_t = V$. For the purposes of this study, we assume that each child order is executed as a *market order* based on the prevailing limit order book structure. We would like our learning agent to modify the AC volume trajectory based on prevailing volume and spread characteristics in the market. As such, the possible actions for our agent include:

- β_j = Proportion of AC_t to trade,
- β_{LB} = Lower bound of volume proportion to trade,
- β_{UB} = Upper bound of volume proportion to trade,
- **Action**: $a_{jt} = \beta_j AC_t$, where $\beta_{LB} \leq \beta_j \leq \beta_{UB}$ and $\beta_j = \beta_{j-1} + \beta_{incr}$.

The aim here is to train the learning agent to trade a higher (lower) proportion of the overall volume when conditions are favourable (unfavourable), whilst still broadly preserving the volume trajectory suggested by the AC model. To ensure that the total volume-to-trade is executed over the given time horizon, we execute any residual volume at the end of the trading period with a *market order*.

5.2.4 Reward function

Each of the actions described above results in a volume to execute with a *market order*, based on the prevailing structure of the limit order book. The size of the child order volume will determine how deep we will need to traverse the order book. For example, suppose we have a BUY order with a *volume-to-trade* of 20 000, split into child orders of 10 000 in period t and 10 000 in period $t+1$. If the structure of the limit order book at time t is as follows:

Market Depth Level	Ask Price	Ask Volume
Level-1	100.00	3000
Level-2	100.50	4000
Level-3	102.30	5000
Level-4	103.00	6000
Level-5	105.50	2000

the volume-weighted execution price will be:

$$\frac{(3000 \times 100) + (4000 \times 100.5) + (3000 \times 102.3)}{10000} = 100.9.$$

Trading more (less) given this limit order book structure will result in a higher (lower) volume-weighted execution price. If the following trading period $t+1$ has the following structure:

Market Depth Level	Ask Price	Ask Volume
Level-1	99.80	6000
Level-2	99.90	2000
Level-3	101.30	7000
Level-4	107.00	3000
Level-5	108.50	1000

the volume-weighted execution price for the second child order will be:

$$\frac{(6000 \times 99.8) + (2000 \times 99.9) + (2000 \times 101.3)}{10000} = 100.12.$$

If the reference price of the stock at $t = 0$ is 99.5, then the *implementation shortfall* from this trade is:

$$\frac{((20000 \times 99.5) - (10000 \times 100.9 + 10000 \times 100.12)}{20000 \times 99.5}$$

$$= -0.0101 = -101bps.$$

Since the conditions of the limit order book were more favourable for BUY orders in period $t + 1$, if we had modified the child orders to, say 8000 in period t and 12000 in period $t + 1$, the resulting *implementation shortfall* would be:

$$\frac{((20000 \times 99.5) - (8000 \times 100.54 + 12000 \times 100.32)}{20000 \times 99.5}$$

$$= -0.0091 = -91bps.$$

In this example, increasing the child order volume when *Ask Prices* are lower and *Level-1 Volumes* are higher decreases the overal cost of the trade. It is for this reason that *implementation shortfall* is a natural candidate for the rewards matrix in our reinforcement learning system. Each action implies a child order volume, which has an associated volume-weighted execution price. The agent will learn the consequences of each action over the trading horizon, with the ultimate goal of minimising the total trade's *implementation shortfall*.

5.2.5 Algorithm

Given the above specification, we followed the following steps to generate our results:

- Specify a stock (S), volume-to-trade (V), time horizon (T), and trading datetime (from which the trading hour H is inferred),
- Partition the dataset into independent *training sets* and *testing sets* to generate results (the *training set* always pre-dates the *testing set*),
- Calibrate the parameters for the Almgren-Chriss (AC) volume trajectory (σ, η) using the historical *training set*; set $\rho = 0$, since we assume order book is resilient to trading activity (see below),
- Generate the AC volume trajectory $(AC_1, ..., AC_T)$,
- Train the *Q-matrix* based on the state-action tuples generated by the *training set*,
- Execute the AC volume trajectory at the specified trading datetime (H) on each day in the *testing set*, recording the *implementation shortfall*,

- Use the trained *Q-matrix* to modify the AC trajectory as we execute V at the specified trading datetime, recording the *implementation shortfall* and
- Determine whether the reinforcement learning (RL) model improved/worsened realised *implementation shortfall*.

In order to train the *Q-matrix* to learn the optimal policy mapping, we need to traverse the training data set ($T \times I \times A$) times, where A is the total number of possible actions. The following pseudo-code illustrates the algorithm used to train the *Q-matrix*:

Algorithm 1 Simple RL Q-learner

```
procedure OPTIMAL_STRATEGY(V, T, I, A)
    for Episode 1 to N do
        Record reference price at t = 0
        for t = T to 1 do
            Calculate episode's STATE attributes (s, v)
            for a = 1 to A do
                Determine the action volume a and resulting remaining inventory i
                Set x = (t, i, s, v)
                Calculate IS from trade, R(x, a)
                Simulate transition x to y
                Lookup max_p Q(y, p)
                Update Q(x, a) = Q(x, a) + αU
            end for
        end for
    end for
    Select the lowest-IS action max_b Q(y, b) for optimal policy
end procedure
```

An important assumption in this model specification is that our trading activity does not affect the market attributes. Although temporary price impact is incorporated into execution prices via depth participation of the *market order* in the prevailing limit order book, we assume the limit order book is resilient with respect to our trading activity. Market resiliency can be thought of as the number of quote updates before the market's spread reverts to its competitive level. Degryse et al. showed that a pure limit order book market (Euronext Paris) is fairly resilient with respect to most order sizes, taking on average 50 quote updates for the spread to normalise following the most aggressive orders [69]. Since we are using 5-minute trading intervals and small trade sizes, we will assume that any permanent price impact effects dissipate by the next trading period. A preliminary analysis of South African stocks revealed that there were on average over 1000 quote updates during the 5-minute trading intervals and the pre-trade order book equilibrium is restored within 2 minutes for large trades. The validity of this assumption however will be tested in future research, as well as other model specifications explored which incorporate permanent effects in the system configuration.

5.3 Data and results

5.3.1 Data used

For this study, we collected 12 months of market depth tick data (Jan-2012 to Dec-2012) from the Thomson Reuters Tick History (TRTH) database, representing a universe of 166 stocks that make up the South African local benchmark index (ALSI) as at 31-Dec-2012. This includes 5 levels of order book depth (bid/ask prices and volumes) at each tick. The raw data was imported into a MongoDB database and aggregated into 5-minute intervals showing average level prices and volumes, which was used as the basis for the analysis.

5.3.2 Stocks, parameters and assumptions

To test the robustness of the proposed model in the South African (SA) equity market we tested a variety of stock types, trade sizes and model parameters. Due to space constraints, we will only show a representative set of results here that illustrate the insights gained from the analysis. The following summarises the stocks, parameters and assumptions used for the results that follow:

- **Stocks**
  - SBK (Large Cap, Financials)
  - AGL (Large Cap, Resources)
  - SAB (Large Cap, Industrials)
- **Model Parameters**
  - β_{LB}: 0, β_{UB}: 2, β_{incr}: 0.25
  - λ: 0.01, τ: 5-min, α_0: 1, γ: 1
  - V: 100 000, 1000 000
  - T: 4 (20-min), 8 (40-min), 12 (60-min)
  - H: 9, 10, 11, 12, 13, 14, 15, 16
  - I, B, W: 5, 10
  - Buy/Sell: BUY
- **Assumptions**
  - Max volume participation rate in order book: 20%

– Market is resilient to our trading activity

Note, we set $\gamma = 1$ since Garcia and Ndiaye state that this is a necessary condition to ensure convergence to the optimal policy with probability one for a finite-horizon MDP [96] (see Chapter 3). We also choose an arbitrary value for λ, although sensitivities to these parameters will be explored in future work. AC parameters are calibrated and *Q-matrix* trained over a 6-month *training set* from 1-Jan-2012 to 30-Jun-2012. The resultant AC and RL trading trajectories are then *executed* on each day at the specified trading time H in the *testing set* from 1-Jul-2012 to 20-Dec-2012. The *implementation shortfall* for both models is calculated and the difference recorded. This allows us to construct a distribution of *implementation shortfall* for each of the AC and RL models, and for all trading hours $H = 9, 10, ..., 16$.

5.3.3 Results

Table 5.1 shows the average % improvement in median *implementation shortfall* for the complete set of stocks and parameter values. These results suggest that the model is more effective for shorter trading horizons ($T = 4$), with an average improvement of up to 10.3% over the base AC model. This result may be biased due to the assumption of order book resilience. Indeed, the efficacy of the trained Q-matrix may be less reliable for stocks which exhibit slow order book resilience, since permanent price effects would affect the state space transitions. In future work, we plan to relax this order book resilience assumption and incoporate permanent effects into state transitions.

Figure 5.1 illustrates the improvement in median post-trade *implementation shortfall* when executing the volume trajectories generated by each of the models, for each of the candidate stocks at the given trading times. In general, the RL model is able to improve (lower) ex-post *implementation shortfall*, however the improvement seems more significant for early morning/late afternoon trading hours. This could be due to the increased trading activity at these times, resulting in more state-action visits in the *training set* to refine the associated Q-matrix values. We also notice more dispersed performance between 10:00 and 11:00. This time period coincides with the UK market open, where global events may drive local trading activity and skew results, particularly since certain SA stocks are dual-listed on the London Stock Exchange (LSE). The improvement in *implementation shortfall* ranges from 15 bps (85.3%) for trading 1000 000 of SBK between 16:00 and 17:00, to -7 bps (-83.4%) for trading 100 000 SAB between 16:00 and 17:00. Overall, the RL model is able to improve *implementation shortfall* by 4.8%.

Figure 5.2 shows the % of *correct actions* implied by the Q-matrix, as it evolves through the training process after each tuple visit. Here, a *correct action* is defined as a reduction

Parameters			Trading Time(hour)								Average
V	T	I,B,W	9	10	11	12	13	14	15	16	
100000	4	5	23.9	-1.4	4.7	13.4	1.8	3.3	1.8	35.1	10.3
100000	8	5	25.3	4.3	8.3	2.3	1.4	9.9	-0.6	-1.9	6.1
100000	12	5	32.7	-25.2	7.2	-2.7	-1.5	4.6	4.5	-3.3	2.1
1000000	4	5	23.3	-1.3	4.8	9.3	1.9	3.5	1.8	35.0	9.8
1000000	8	5	28.8	5.6	8.2	1.9	1.4	9.9	-0.3	-2.6	6.6
1000000	12	5	33.1	-25.0	7.2	-4.0	-0.8	4.8	4.8	1.2	2.7
100000	4	10	22.9	1.3	3.0	9.7	2.7	5.8	3.5	-26.1	2.8
100000	8	10	26.0	4.3	6.7	-0.2	3.5	8.6	1.6	-3.1	5.9
100000	12	10	27.8	-21.9	7.5	-4.1	0.6	1.8	6.2	-9.5	1.1
1000000	4	10	22.6	1.4	3.1	9.3	2.5	6.0	3.6	-26.1	2.8
1000000	8	10	26.3	5.0	7.2	-0.5	3.3	7.0	2.3	-1.8	6.1
1000000	12	10	27.9	-24.3	8.3	-6.9	0.5	1.8	7.5	-3.3	1.4

TABLE 5.1: Average % improvement in median *implementation shortfall* for various parameter values, using AC and RL models. Training H-dependent.

Parameters			Standard Deviation(%)		% improvement
V	T	I,B,W	AC	RL	in IS
100000	4	5	0.13	0.17	10.3
100000	8	5	0.14	0.23	6.1
100000	12	5	0.14	0.26	2.1
1000000	4	5	0.13	0.17	9.8
1000000	8	5	0.14	0.23	6.6
1000000	12	5	0.14	0.26	2.7
100000	4	10	0.13	0.17	2.8
100000	8	10	0.14	0.22	5.9
100000	12	10	0.14	0.26	1.1
1000000	4	10	0.13	0.17	2.8
1000000	8	10	0.14	0.22	6.1
1000000	12	10	0.14	0.26	1.4
Average			0.14	0.22	4.8

TABLE 5.2: Standard deviation(%) of implementation shortfall when using AC vs RL models.

(addition) in the volume-to-trade based on the max Q-value action, in the case where *spreads* are above (below) the 50%ile and *volumes* are below (above) the 50%ile level. This coincides with the intuitive behaviour we would like the RL agent to learn. These results suggest that finer state granularity ($I, B, W = 10$) improves the overall accuracy of the learning agent, as demonstrated by the higher % *correct actions* achieved. All model configurations seem to converge to some *stationary* accuracy level after approximately 1000 tuple visits, suggesting that a shorter training period may yield similar results. We do however note that improving the % of correct actions by increasing the

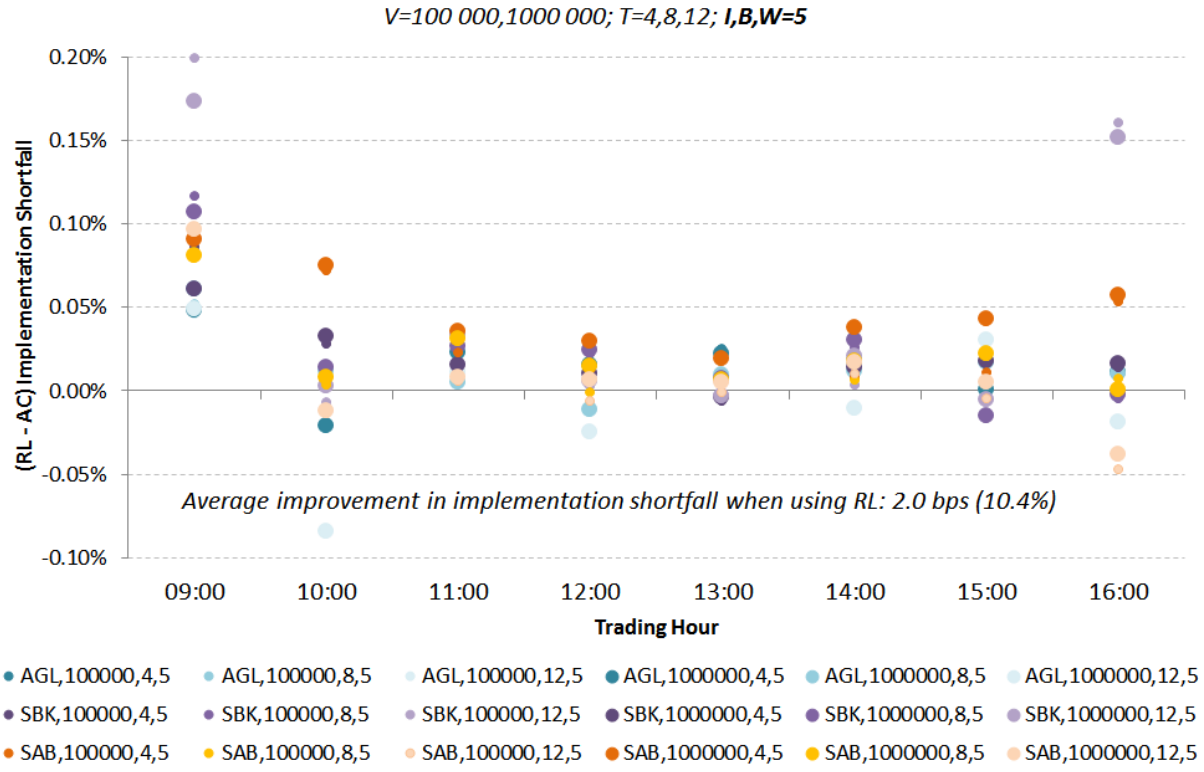


FIGURE 5.1: Difference between median implementation shortfall generated using RL and AC models, with given parameters ($I,B,W = 5$). Training H-dependent.

granularity of the state space does not necessarily translate into better model performance. This can be seen by Table 5.1, where the results where $I, B, W = 10$ do not show any significant improvement over those with $I, B, W = 5$. This suggests that the market dynamics may not be fully represented by *volume* and *spread* state attributes, and alternative state attributes, such as *volume imbalance* and *quote depth*, should be explored in future work to improve ex-post model efficacy.

Table 5.2 shows the average standard deviation of the resultant *implementation shortfall* when using each of the AC and RL models. Since we have not explicitly accounted for *variance of execution* in the RL reward function, we see that the resultant trading trajectories generate a higher standard deviation compared to the base AC model. Thus, although the RL model provides a performance improvement over the AC model, this is achieved with a higher degree of execution risk, which may not be acceptable for the trader. We do note that the RL model exhibits comparable risk for $T = 4$, thus validating the use of the RL model to reliably improve IS over short trade horizons. A future refinement on the RL model should incorporate *variance of execution*, such

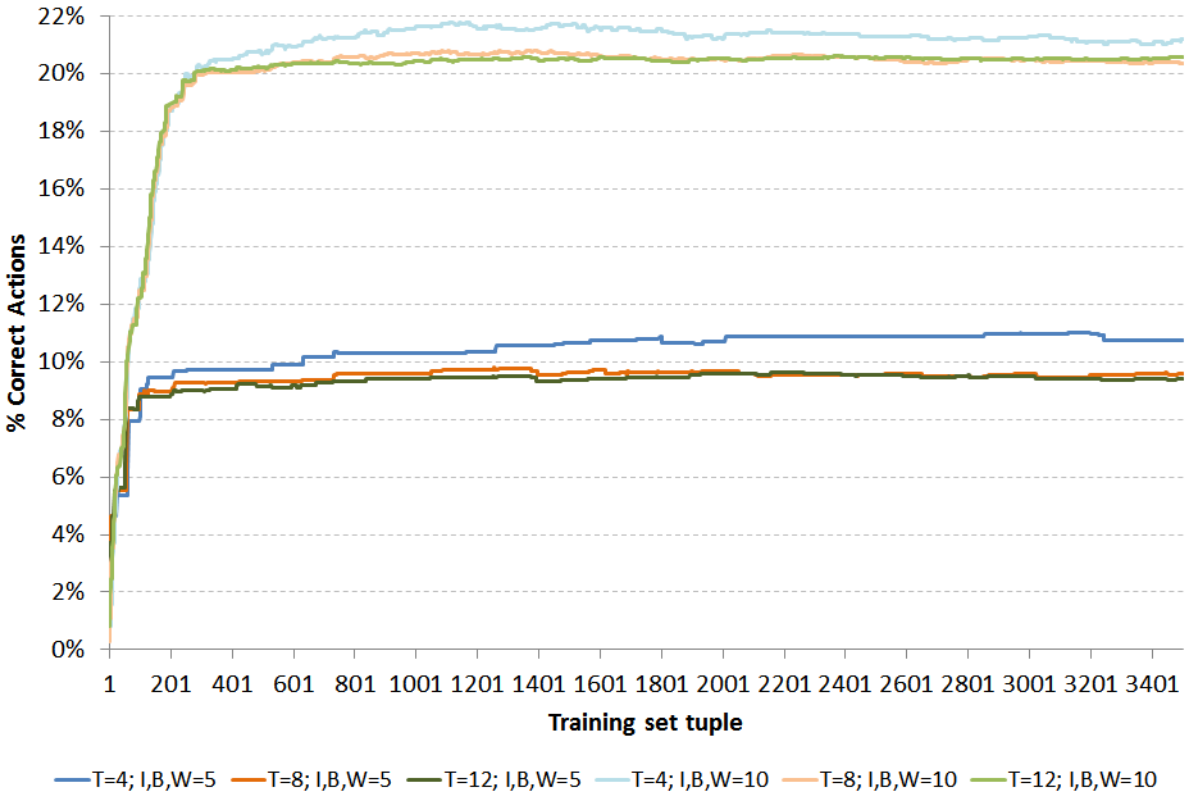


FIGURE 5.2: % correct actions implied by Q-matrix after each training set tuple. Training H-dependent.

that it is consistent with the AC objective function. In this way, a true comparison of the techniques can be done, and one can conclude as to whether the RL model indeed outperforms the AC model at a statistically significant level.

5.4 Some remarks

In this chapter, we introduced reinforcement learning as a candidate machine learning technique to *enhance* a given optimal liquidation volume trajectory. Nevmyvaka, Feng and Kearns showed that reinforcement learning delivers promising results where the learning agent is trained to choose the optimal limit order price at which to place the remaining inventory, at discrete periods over a fixed liquidation horizon [191, 192]. Here, we show that reinforcement learning can also be used successfully to modify a given volume trajectory based on market attributes, executed via a sequence of *market orders* based on the prevailing limit order book.

Specifically, we showed that a simple look-up table *Q-learning* technique can be used to train a learning agent to modify a static Almgren-Chriss volume trajectory based on

prevailing spread and volume dynamics, assuming order book resiliency. Using a sample of stocks and trade sizes in the South African equity market, we were able to reliably improve post-trade *implementation shortfall* by up to 10.3% on average for short trade horizons, demonstrating promising potential applications of this technique. Further investigations include incorporating *variance of execution* in the RL reward function, relaxing the order book resiliency assumption and alternative state attributes to govern market dynamics.

We note that the model presented here requires pre-specification of system features (spread, volume) to serve as public attributes in the chosen state space of the learning agent. In addition, each feature requires discretisation at a specified resolution to make learning feasible in finite time. While these features may capture salient properties of the temporal evolution of the LOB, we conjecture that there is a richer representation which better captures the nuances of the complex behaviour of the system, utilising the lens of the machine trading agent to capture scale-specific dynamics. In the chapters which follow, we aim to make these ideas more concrete. In Chapter 6, we develop a technique for unsupervised, offline estimation of a public state attribute which captures the (exogenous) scale-specific evolution of the complex system, and we provide a scheme to detect the state online. The efficacy of this state representation is tested in Chapter 8. In Chapter 9, we develop a scheme for unsupervised, online enumeration of the agent's state space *at the scale of interaction*. This not only allows an agent to make sense of an asynchronous market data feed and learn optimal trading policies over time, but also provides a scheme to encode the effect of agent interactions on the state space it perceives - a key property of complex adaptive systems.

Chapter 6

Detecting intraday states from streaming market microstructure features

6.1 Overview

This chapter considers the refinement of the state representation for the reinforcement learning agent, using the complex system ideology discussed in Chapter 1 to inform a unique approach for extracting persistent intraday temporal dynamics from a streaming market data feed, as well as a scheme for *online state detection* to enable online learning. The contribution is captured in the following two papers:

D. Hendricks, T. Gebbie, D. Wilcox. *High-speed detection of emergent market clustering via an unsupervised parallel genetic algorithm.* South African Journal of Science, vol. 112, no. 1/2, 2016. [125]
Available online: http://dx.doi.org/10.17159/sajs.2016/20140340

D. Hendricks, T. Gebbie, D. Wilcox. *Detecting intraday financial market states using temporal clustering.* Quantitative Finance, 2016.
Available online: http://dx.doi.org/10.1080/14697688.2016.1171378 [126]

6.2 From unsupervised clustering to temporal states

We consider the use of a physical analogy to the ferromagnetic Potts model at thermal equilibrium to describe object interactions, before deriving an unsupervised clustering

algorithm, where both the number of clusters and configuration emerges from the data [38, 39, 103, 249]. Treating intraday time periods as objects, the algorithm will be used to identify intraday market states from observed market microstructure features. Although Marsili used a similar approach to classify days as states [172], the authors are unaware of another study which applies this technique to *intraday* period clustering using *multiple* features. In addition, a high-speed Parallel Genetic Algorithm (PGA) will be used for efficient computation of the cluster configurations, with absolute computation speeds conducive to overnight or even intraday recalibration of identified states [125].

The results reveal an interesting hierarchy of system behaviour at different time scales. Statistically significant power-law fits to configuration characteristics suggest scale-invariant behaviour which may translate to persistent features in market states. In addition, the power-law fits yield different scaling exponents at the different time scales, suggesting the existence of different universality classes characterising behaviour at each scale [62, 78, 94]. This motivates the importance of time-scale specific information when planning in this domain. Here we are considering a particular case of *calendar time* when investigating scale-related phenomena. There is a rich history in the literature which has aimed to directly model the *event time* foundations of market microstructure processes. The seminal work of Garman [97], which used point processes to model order book events, forms the basis of many subsequent *event time* approaches to modelling transaction and quote data. An important extension of this view is the vector autoregressive model for trades and quotes developed by Hasbrouck [116, 117] and Engle and Russell [79]. A complementary approach introduces the concept of *intrinsic time*, which aims to measure trading opportunities in reference to specific features of traded stocks, for example, using the rate of trading to modify calendar or chronological time. These are discussed by Müller et al. [187] and Derman [71]. The more recent use of Hawkes processes to model mutually-exciting order book events [3, 24, 159, 232] is an important return to the idea of viewing events as a foundational concept when modelling transactions and order book dynamics.

Easley et al. introduce the *volume time* paradigm for high-frequency trading, with the clock ticking according to the number of events (proxied by trade volume) flowing through the system [76]. This is a pragmatic attempt to reconcile the foundational event-based paradigm introduced by Garman [97] with the wide use of chronological or calendar time. They argue that machines operate on a clock which is not chronological, but rather related to the number of cycles per instruction initiated by an event [76, 208]. This allows one to measure time in terms of frequency of changes in information, as measured by trading volumes. When one considers the *complex event processing* paradigm which underpins many automated trading systems in financial markets [5], one can appreciate the suitability of the event-based clock and the view that the calendar time clock is

a legacy convenience from the low-frequency, human-trader-driven world. As the shift from human-driven to machine-driven trading dominates financial markets, the study of event-time-scale phenomena has become increasingly important and warrants further exploration.

While the identified market states reveal many interesting insights, trading agents would benefit from being able to detect online (or in real-time) which state they are *currently* in. We develop a novel technique which extracts the characteristic signature of market activity from each of the identified states, and uses this as the basis for an online state detection algorithm. In one application, this is used to construct 1-step transition probability matrices, which can be refined online and used in optimal planning algorithms.

6.3 Super-paramagnetic clustering for state discovery and detection

Blatt et al. proposed a novel non-parametric clustering approach, based on an analogy to the ferromagnetic Potts model at thermal equilibrium [38, 39, 249]. By assigning a Potts spin variable to each object and introducing a short-range distance-dependent ferromagnetic interaction field, regions of aligned spins emerge, which are analogous to groups of objects in the same cluster, where *spin alignment* suggests *object homogeneity* [242].

6.3.1 Potts spin models as analogue for financial system

One can apply super-paramagnetic ordering of a q-state Potts model directly for cluster identification [38]. In a market Potts model, each stock can take on q-states and each state can be represented by a cluster of similar stocks [38, 103, 154]. Cluster membership is indicative of some commonality among the cluster members. Each stock has a component of its dynamics as a function of the state it is in and a component of its dynamics influenced by stock specific noise. In addition, there may be global couplings that influence all the stocks, i.e. the external field that represents a market mode.

More formally, consider a q-state Potts model with spins $s_i = 1, ..., q$ for $i = 1, ..., N$, where N is the total number of objects in the system. The cost function is given by the following Hamiltonian:

$$H = -\sum_{s_i, s_j \in \mathcal{S}} J_{ij} \delta(s_i, s_j) \tag{6.1}$$

where the spins s_i can take on q-states and the coupling of the ith and j^{th} object are governed by J_{ij}. In the case of object clustering for a data sample, a candidate configuration is given by the set $\mathcal{S} = \{s_i\}_{i=1}^{N}$, where s_i represents the cluster group index to which the ith object belongs. One can consider the coupling parameters J_{ij} as being a function of the correlation coefficient C_{ij} [103, 154]. This is used to specify a distance function that is decreasing with distance between objects. If all the spins are related in this way, then each pair of spins is connected by some non-vanishing coupling $J_{ij} = J_{ij}(C_{ij})$. This allows one to interpret s_i as a Potts spin in the Potts model Hamiltonian with J_{ij} decreasing with the distance between objects [38, 154]. The case where there is only one cluster can be thought of as a ground state. As the system becomes more excited, it could break up into additional clusters. Each cluster would have specific Potts magnetisations, even though the nett magnetisation can be zero for the complete system. Generically, the correlation would then be both a function of time and temperature in order to encode both the evolution of clusters, as well as the hierarchy of clusters as a function of temperature. In the basic approach, one is looking for the lowest energy state that fits the data.

6.4 A maximum likelihood approach

In order to parameterise the model efficiently, one can choose to make an ansatz for the data generative function [193] and use this to develop a maximum-likelihood approach [103], rather than explicitly solving the Potts Hamiltonian numerically [38, 154]. A number of authors have considered this approach for object clustering [103, 179, 188], however we follow the proposition by Giada and Marsili [103]. A summary exposition will be presented here (as shown in [125, 126]), with a full derivation available in the Appendices.

According to the Noh ansatz [193], the generative model of the time series associated with the ith object can then be written as

$$x_i(t) = g_{s_i}\eta_{s_i} + \sqrt{1 - g_{s_i}^2}\epsilon_i \tag{6.2}$$

where the cluster-related influences are driven by η_{s_i} and the object-specific effects by ϵ_i, both treated as Gaussian random variables with unit variance and zero mean[1]. The

[1]This form of the price model ensures that the self correlation of a stock is one and independent of the cluster coupling. This can be seen by computing the self correlation $E[x_i^2]$ and using that clusters and stock unique process are unit variance zero mean processes

$$\mathrm{E}[(g_{s_i}\eta_{s_i} + \sqrt{1 - g_{s_i}^2}\epsilon_i)^2] = g_{s_i}^2 + (1 - g_{s_i}^2) = 1. \tag{6.3}$$

relative contribution is controlled by the intra-cluster coupling parameter g_{s_i}. The Noh-Giada-Marsili model encodes the idea that objects which have something in common belong in the same cluster, object membership in a particular cluster is mutually exclusive and intra-cluster correlations are positive.

If one takes Equation 6.2 as a statistical hypothesis, it is possible to compute the probability density $P(\{\bar{x_i}\}|\mathcal{G},\mathcal{S})$ for any given set of parameters $(\mathcal{G},\mathcal{S}) = (\{g_s\},\{s_i\})$ by observing the data set $\{\bar{x}_i\}, i = 1,...,N$ as a realisation of the common component of Equation 6.2 as follows [103]:

$$P(\{\bar{x_i}\}|\mathcal{G},\mathcal{S}) = \prod_{d=1}^{D}\left\langle \prod_{i=1}^{N} \delta\left(x_i(d) - (g_{s_i}\eta_{s_i} + \sqrt{1-g_{s_i}^2}\epsilon_i)\right)\right\rangle. \tag{6.5}$$

In Equation 6.5, N is the number of objects and D is the number of feature measurements for each object. The variable δ is the Dirac delta function and $\langle ... \rangle$ denotes the average over all permissible values for η_{s_i} and ϵ_i. Note that the presence of a bar, i.e. $\{\bar{x_i}\}$, indicates a sample of *observed* time series values for object i. For a given cluster structure $\mathcal{S}$, the likelihood is maximal when the parameter g_s takes the values

$$g_s^* = \begin{cases} \sqrt{\frac{c_s - n_s}{n_s^2 - n_s}} & \text{for } n_s > 1, \\ 0 & \text{for } n_s \leq 1. \end{cases} \tag{6.6}$$

n_s in Equation 6.6 denotes the number of objects in cluster s, i.e.

$$n_s = \sum_{i=1}^{N} \delta_{s_i,s}. \tag{6.7}$$

The variable c_s is the internal correlation of the sth cluster, denoted by the following equation:

$$c_s = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}\delta_{s_i,s}\delta_{s_j,s}. \tag{6.8}$$

The variable C_{ij} is the *Pearson correlation coefficient* of the data, denoted by the following equation:

$$C_{ij} = \frac{\bar{x_i}\bar{x_j}}{\sqrt{\|\bar{x_i}^2\|\|\bar{x_j}^2\|}}. \tag{6.9}$$

This is not a unique choice, another possible choice often used is

$$\mathrm{E}[(\frac{\sqrt{g_{s_i}}}{\sqrt{1+g_{s_i}}}\eta_{s_i} + \frac{1}{\sqrt{1+g_{s_i}}}\epsilon_i)^2] = \frac{1+g_{s_i}}{1+g_{s_i}} = 1. \tag{6.4}$$

The maximum likelihood of structure $\mathcal{S}$ can be written as $P\left(\mathcal{G}^*, \mathcal{S}|\bar{x}_i\right) \propto \exp^{D\mathcal{L}_c(\mathcal{S})}$, where the resulting likelihood function per feature $\mathcal{L}_c$ is denoted by

$$\mathcal{L}_c(\mathcal{S}) = \frac{1}{2} \sum_{s:n_s>1} \left(\log \frac{n_s}{c_s} + (n_s - 1) \log \frac{n_s^2 - n_s}{n_s^2 - c_s} \right). \tag{6.10}$$

From Equation 6.10, it follows that $\mathcal{L}_c = 0$ for clusters of objects that are uncorrelated, i.e. where $g_s^* = 0$ or $c_s = n_s$ or when the objects are grouped in singleton clusters for all the cluster indexes ($n_s = 1$). Equations 6.8 and 6.10 illustrate that the resulting maximum likelihood function for $\mathcal{S}$ depends on the *Pearson correlation coefficient* C_{ij} and hence exhibits the following advantages in comparison to conventional clustering methods:

- It is **unsupervised**: The optimal number of clusters is unknown *a priori* and not fixed at the outset
- The interpretation of results is **transparent** in terms of the model, namely Equation 6.2.

Giada and Marsili state that $\max_s \mathcal{L}_c(\mathcal{S})$ provides a measure of structure inherent in the cluster configuration represented by the set $\mathcal{S} = \{s_1, ..., s_N\}$ [103]. The higher the value, the more pronounced the structure. A full derivation confirming the likelihood function proposed by Giada and Marsili in Equation 6.10 can be found in a paper by Hendricks et al. [126], and is reproduced here in Appendix A.

We note that the particular choice of Gaussian innovations in Equation 6.2 is convenient, since the *Pearson correlation coefficient* then completely characterises pairwise interactions amongst objects in the system [103]. This is a necessary condition, given the physical analogy and link to the motivating Hamiltonian given in Equation 6.1. The application of this technique to high-frequency financial time series may motivate a more prudent assumption for the underlying object and cluster dynamics, incorporating jumps to better model the price formation process at this scale. However, the use of, say, jump diffusion innovations would require an alternative dependency metric, such as Lévy copulas, to completely capture object interactions [61, 180], requiring a careful re-derivation of the appropriate likelihood function. This will be explored in further research.

6.5 Considering time periods as objects for market state determination

The data generative model specified by Equation 6.2 is sufficiently generic that it can be applied to a diverse set of problem domains, where object and cluster innovations can be assumed to be Gaussian. In the financial domain, initial applications focused on clustering stocks based on price changes [103, 125, 126], however Marsili proposed that this technique could be used to cluster *time periods* in order to identify *temporal market states* [172]. Days were grouped into clusters based on the closing price performance of the chosen universe of stocks, demonstrating a meaningful classification of market-wide activity which persists through time [172]. We propose that a similar approach can be applied to discover *intraday* temporal states, clustering time periods based on the performance of *multiple* observable market microstructure features. A practical trading system often has access to a real-time market data feed, from which multiple features can be extracted to describe various aspects of the evolving limit order book. In addition, examining temporal cluster configurations at varying time scales can suggest a hierarchy of system behaviour, providing insights into exogenous and endogenous market activity. This can also assist trading agents in developing optimal trajectories for varying objectives, such as stock acquisition or liquidation at minimal cost. In particular, for an agent tasked to learn an optimal policy (state-action mapping), the grouping of temporal periods into market states based on market microstructure feature performance provides a novel scheme to reduce the dimensionality of the state space and promote efficient learning.

In this chapter, we will focus on the emergent hierarchy of system behaviour at different time scales and explore a scheme for online state detection. In one application, this leads to a system of 1-step state transition probability matrices at varying scales, which can be refined online in real-time. These can be used in optimal planning schemes where Markovian dynamics are assumed and state persistence can be exploited.

6.6 State Signature Vectors for online state detection

Recall that the model presented in Chapter 5 required pre-specification of system features (spread, volume) to serve as public attributes in the chosen state space of the learning agent, as well as discretisation at a specified resolution to make learning feasible in finite time. The scheme presented in this chapter allows us to enumerate a scale-specific state space by grouping temporal periods based on feature performance. We can develop these ideas further to ensure the identified states can be detected online,

and reduce the state space to those which are most likely to persist. This can then be incorporated as a rich public state attribute for the state space of the trading agent to promote efficient, but effective learning.

The clustering procedure described thus far can be used as an unsupervised algorithm to group temporal periods into states according to feature similarity, however this can only reveal the *ex-ante* temporal states and is not suitable for online detection. Upon examination of the resulting cluster configurations, we noted that each node refers to a particular time period, with an associated signature of market activity. Furthermore, if two time periods appear in the same cluster, given the data generative model assumed in Equation 6.2, we conjecture that it is the relative similarity of their characteristic signatures of market activity which resulted in their assignment to the same cluster. Using this idea, given a cluster configuration of temporal periods into market states, it is possible to extract a *state signature vector* (SSV) which summarises the signature of market activity across stocks and time periods for each state. Then, if one is faced with a new candidate feature vector (FV), the market state assignment can be determined by using the closest match within the set of pre-determined SSVs computed offline. FVs are easy to compute online from a streaming datafeed and state assignment can be achieved using a simple Euclidean distance computation. To make these ideas concrete, consider the example illustrated in Figure 6.1.

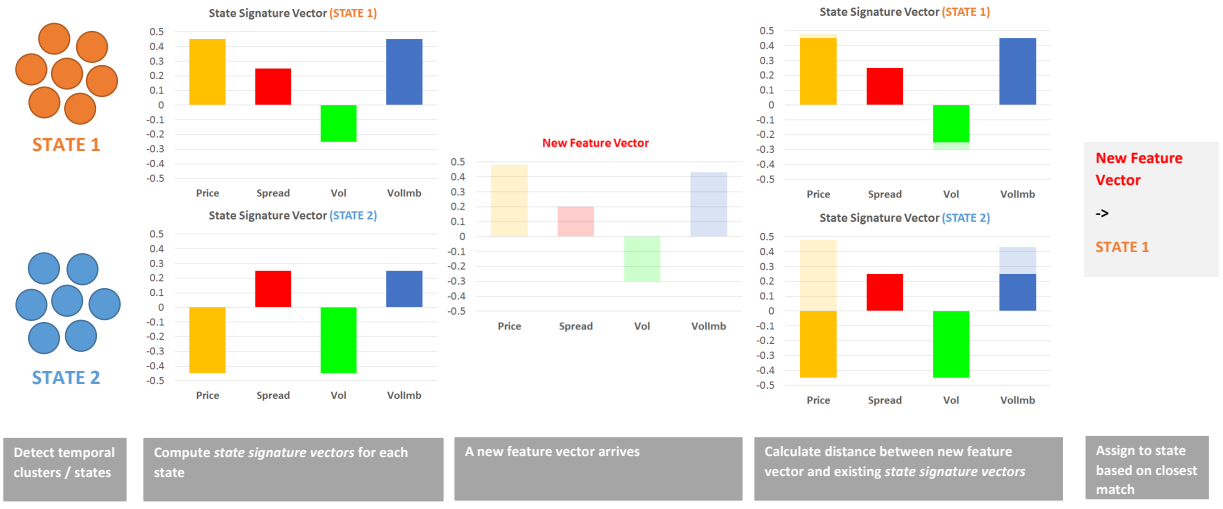


FIGURE 6.1: Illustration of online state assignment based on identified state signature vectors.

Here, we compute two SSVs from the identified states, and use these as a basis for assigning a new FV to a market state. This is based on a simple Euclidean distance metric,

$$\text{argmin}_p ||FV - SSV_p||,$$

where p is the index of the identified states.

We have used four features to characterise market activity at intraday scale. These include: *trade price, trade volume, spread* and *quote volume imbalance*. In particular, we consider the *relative change* in each of these features. For example, based on a set of feature measurements $\mathcal{F}^{5min}$ at 5-minute scale, we would compute

$$\triangle f_t^{5min} = \frac{f_t^{5min} - f_{t-1}^{5min}}{f_{t-1}^{5min}}$$

for all $f_t^{5min} \in \mathcal{F}^{5min}$. For the initial temporal cluster detection stage, these "feature returns" are calculated for each stock and concatenated before computing the time period correlation matrix.

For the extraction of SSVs from significant states, we compute average feature returns across member periods and stocks. Although this results in a loss of information, we conjecture that the average signature of feature returns broadly captures the state of market activity. The SSVs for each time-scale configuration are illustrated in Figures 6.5, 6.7, 6.9 and 6.11. Following this approach, the FVs calculated in the online environment would constitute the same averages of feature returns, before matching to the appropriate SSV. Alternative schemes for extraction of SSVs which preserve state-specific information will be explored in future work. The chosen features do not represent an exhaustive set of possible explanatory factors for intraday market activity, but rather were chosen based on the relative ease of their online construction from streaming Level-1 market data feeds [146]. Additional features can be considered in future work.

6.7 Scale-invariant characteristics of states

The detected temporal cluster configurations can be further analysed to determine whether any characteristics exhibit scale-invariant behaviour. In particular, a visual inspection of the cluster configurations shown in Section 6.9.4 led us to conjecture a possible power-law fit for cluster sizes. Many physical and man-made systems exhibit characteristics which follow a power-law functional form, and its unique mathematical properties sometimes lead to surprising physical insights [58, 94]. Many authors have investigated the nature of information and forecasting at different time scales in financial markets (see [62, 78, 255] as examples). For our application, the existence of different critical exponents for the best power-law fits at different time scales may suggest different universality classes which characterise the system activity at each scale. In fact, Mastromatteo and Marsili [175] discuss the notion that, for a complex adaptive system, distinguishable models can only be gleaned when the system is near criticality. Thus, if

financial markets truly are a complex adaptive system, measurable quantities from the dynamics at each scale should yield a statistically significant power-law fit. Although it is difficult to quantify the exact nature of these scale-specific behaviours or universality classes, their apparent existence suggests that investment and trading decisions would benefit from time-scale-specific state space information. This would enhance the efficacy of intraday policies which aim to find optimal trajectories through the system.

Given the difficulties of identifying statistically significant power-law fits to empirical quantities [28], we incorporated the maximum likelihood fitting procedure provided by Clauset, Shalizi and Newman [58]. Outputs from their functions include the scaling parameter of the proposed power-law fit, a Kolmogorov-Smirnov test for the goodness-of-fit of the proposed model to the data, the lower-bound for the fit if the tail distribution follows a power-law and the log-likelihood of the data under the power-law fit.

We note that a detected temporal cluster configuration results in a set of homogeneous market states, although it is not clear which states are significant, i.e. likely to persist, or merely transient. Using *all* identified states may result in spurious state assignments if one uses the online algorithm described in Section 6.6. This leads to the need for some selection criteria for significant states, before extracting SSVs. Candidate criteria include using intra-cluster connectedness (c_s) or cluster size with some form of thresholding procedure, however these heuristics are inherently subjective. The power-law fit to cluster size provides one candidate objective approach for state selection. By selecting the clusters which satisfy the power-law functional form, we conjecture that the scale invariant properties of this fit imply persistent properties for temporal market states, resulting in an objective mechanism for selecting significant states. This reduces the set of SSVs which form the basis for the online state detection algorithm.

6.8 A high-speed Parallel Genetic Algorithm implementation

In this section, we introduce a maintainable and scalable master-slave parallel genetic algorithm (PGA) framework for unsupervised cluster analysis on the CUDA platform, which is able to detect clusters using the Giada and Marsili likelihood function. By applying the proposed cluster analysis approach and examining the clustering behaviour of financial instruments, this offers a unique perspective to monitoring the intraday characteristics of the stock market and the detection of structural changes in near-real-time. The novel implementation presented here builds on the contribution of Cieslakiewicz [55]. While this chapter provides an overview and specific use-case for the algorithm,

the authors are investigating aspects of adjoint parameter tuning, performance scalability and the impact on solution quality for varying stock universe sizes and cluster types.

In order to localise clusters of normalised stock returns in financial data, Giada and Marsili made use of a *simulated annealing* algorithm [103, 104], with $-\mathcal{L}_c$ as the cost function for their application of the log-likelihood function on real-world data sets to substantiate their approach. This was then compared to other clustering algorithms, such as *K-means*, *single linkage*, *centroid linkage*, *average linkage*, *merging* and *deterministic maximisation* [104]. The technique was successfully applied to South African financial data by Mbambiso et al., using a serial implementation of a *simulated annealing* algorithm (see [177] and [101]).

Simulated annealing and *deterministic maximisation* provided acceptable approximations to the maximum likelihood structure, but were inherently computationally expensive. We promote the use of PGAs as a viable approach to approximate the maximum likelihood structure. The likelihood function, $\mathcal{L}_c$, will be used as the fitness function and a PGA algorithm will be used to find the maximum for $\mathcal{L}_c$, in order to efficiently isolate clusters in correlated financial data.

6.8.1 GA principle and genetic operators

One of the key advantages of GAs is that they are conceptually simple. The core algorithm can be summarised into the following steps: *initialise population*, *evolve individuals*, *evaluate fitness*, *select individuals to survive to the next generation*. GAs exhibit the trait of broad applicability [223], as they can be applied to any problem whose solution domain can be quantified by a function which needs to be optimised.

Specific genetic operators are applied to the parents, in the process of reproduction, which then give rise to offspring. The genetic operators can be classified as follows:

Selection: The purpose of selection is to isolate fitter individuals in the population and allow them to propogate in order to give rise to new offspring with higher fitness values. We implemented the *stochastic universal sampling selection operator*, where individuals are mapped to contiguous segments on a line in proportion to their fitness values [25]. Individuals are then selected by sampling the line at uniformly spaced intervals. While fitter individuals have a higher probability of being selected, this technique improves the chances that weaker individuals will be selected, allowing diversity to enter the population and reducing the probability of convergence to a local optimum.

Crossover: Crossover is the process of mating two individuals, with the expectation that they can produce a fitter offspring [223]. The crossover genetic operation involves the selection of random loci to mark a cross site within the two parent chromosomes, copying the genes to the offspring. A bespoke *knowledge-based crossover* operator was developed for our implementation [55], in order to incorporate domain knowledge and improve the rate of convergence.

Mutation: Mutation is the key driver of diversity in the candidate solution set or search space [223]. It is usually applied after crossover and aims to ensure that genetic information is randomly distributed, preventing the algorithm from being trapped in local minima. It introduces new genetic structures in the population by randomly modifying some of its building blocks and enables the algorithm to traverse the search space globally.

Elitism: Coley states that fitness-proportional selection does not necessarily favour the selection of any particular individual, even if it is the fittest [59]. Thus the fittest individuals may not survive an evolutionary cycle. Elitism is the process of preserving the fittest individuals by inherent promotion to the next generation, without undergoing any of the genetic transformations of crossover or mutation [223].

Replacement: Replacement is the last stage of any evolution cycle, where the algorithm needs to replace old members of the current population with new members [223]. This mechanism ensures that the population size remains constant, while the weakest individuals in each generation are dropped.

Although GAs are very effective for solving complex problems, this positive trait can unfortunately be offset by long execution times, due to the traversal of the search space. GAs lend themselves to parallelisation, provided the fitness values can be determined independently for each of the candidate solutions. While a number of schemes have been proposed in the literature to achieve this parallelisation (see [138], [223] and [211]), we have chosen to implement the *master-slave* model.

6.8.2 Master-slave parallelisation

Master-slave GAs, or global PGAs, involve a single population, but distributed amongst multiple processing units for determination of fitness values and the consequent application of genetic operators. They allow for computation on shared-memory processing entities or any type of distributed system topology, for example grid computing [211].

Ismail provides a summary of the key features of the master-slave PGA [138]: The algorithm uses a single population (stored by the master) and the fitness evaluation of all of the individuals is performed in parallel (by the slaves). Communication occurs only as each slave receives the individual (or subset of individuals) to evaluate and when the slaves return the fitness values, sometimes after mutation has been applied with the given probability. The particular algorithm we implemented is *synchronous*, i.e. the master waits until it has received the fitness values for all individuals in the population before proceeding with selection and mutation. The *synchronous* master-slave PGA thus has the same properties as a conventional GA, except evaluation of the fitness of the population is achieved at a faster rate. The algorithm is relatively easy to implement and a significant speedup can be expected if the communications cost does not dominate the computation cost. The whole process has to wait for the slowest processor to finish its fitness evaluations until the selection operator can be applied.

A number of authors have used the Message Parsing Interface (MPI) paradigm to implement a master-slave PGA. Digalakis and Margaritis implement a synchronous MPI PGA and shared-memory PGA, whereby fitness computations are parallelised and other genetic operators are applied by the master node only [73]. They demonstrate a computation speed-up which scales linearly with the number of processors for large population sizes. Zhang et al. use a centralised control island model to concurrently apply genetic operators to sub-groups, with a bespoke migration strategy using elite individuals from sub-groups [254]. Nan et al. used the MATLAB parallel computing and distributed computing toolboxes to develop a master-slave PGA [189], demonstrating its efficacy on the image registration problem when using a cluster computing configuration.

For our implementation, we made use of the Nvidia CUDA platform to achieve massive parallelism by utilising the Graphical Processing Unit (GPU) Streaming Multiprocessors (SM) as slaves, and the CPU as master.

6.8.3 Computational Platform and Implementation

Compute Unified Device Architecture (CUDA) is Nvidia's platform for massively parallel high performance computing on the Nvidia GPUs. Compute Unified Device Architecture (CUDA) is Nvidia's platform for massively parallel high-performance computing on the Nvidia GPUs. At its core are three key abstractions: a hierarchy of thread groups, shared memories, and barrier synchronisation. Full details on the execution environment, thread hierarchy, memory hierarchy and thread synchronisation schemes have been omitted here, but we refer the reader to Nvidia technical documentation [200, 202] for a comprehensive discussion.

6.8.3.1 Specific computational environment

The CUDA algorithm and the respective testing tools were developed using Microsoft Visual Studio 2012 Professional, with the Nvidia Nsight extension for CUDA-C projects. The configurations listed in Table 6.1 were tested to determine the versatility of the CUDA clustering algorithms on different architectures.

Environment	**Configuration**	**Framework**
GTX_CUDA	Windows 7 Professional Service Pack 1 (64-bit), Intel Core i7-4770K CPU@3.5 GHz, 32GB RAM, Nvidia GTX Titan Black with 6GB RAM, CC: 3.0, SM: 3.5	CUDA 5.5 (parallel)
GTX_MATLAB	Windows 7 Professional Service Pack 1 (64-bit), Intel Core i7-4770K CPU@3.5 GHz, 32GB RAM, Nvidia GTX Titan Black with 6GB RAM, CC: 3.0, SM: 3.5	MATLAB 2013a (serial)
TESLA_CUDA	Windows 7 Professional Service Pack 1 (64-bit), Intel Core i7-X980 CPU@3.33 GHz, 24GB RAM, Nvidia TESLA C2050 with 2.5GB RAM, CC: 2.0, SM: 2.0	CUDA 5.5 (parallel)
TESLA_MATLAB	Windows 7 Professional Service Pack 1 (64-bit), Intel Core i7-X980 CPU@3.33 GHz, 24GB RAM, Nvidia TESLA C2050 with 2.5GB RAM, CC: 2.0, SM: 2.0	MATLAB 2013a (serial)

TABLE 6.1: Development, testing and benchmarking environments

We had the opportunity to test two candidate graphics cards for the algorithm implementation: the Nvidia GTX Titan Black and the Nvidia TESLA C2050. Both cards offer double-precision calculations and a comparable number of CUDA cores and TFLOPS (tera floating point operations per second), however the GTX card is significantly cheaper than the TESLA card. The primary reason for this is the use of ECC (error check and correction) memory on the TESLA cards, where extra memory bits are present to detect and fix memory errors [1]. The presence of ECC memory ensures consistency in results generated from the TESLA card, which is critical for rigorous scientific computing. In further investigations, the authors will explore the consistency of the solution quality generated from the GTX card, and whether the resultant error is small enough to justify the cost saving compared to the TESLA card.

6.8.3.2 Implementation

The following objectives were considered in this research: 1) investigate and tune the behaviour of the PGA implementation using a pre-defined set of 40 simulated stocks featuring 4 distinct disjoint clusters; 2) identify clusters in a real-world dataset, viz. high-frequency price evolutions of stocks; and 3) test the efficiency of the GPU environment.

6.8.3.3 Representation

We used integer-based encoding for the representation of individuals in the genetic algorithm, i.e.

$$Individual = \mathcal{S} = \{s_1, s_2, ..., s_{i-1}, s_i, ..., s_N\} \tag{6.11}$$

where $s_i = 1, ..., q$ and $i = 1, ..., N$. Here, s_i is the cluster that object i belongs to. In terms of the terminology pertaining to GAs, it means that the ith gene denotes the cluster that the ith object or asset belongs to. The numbers of objects or assets is N, thus to permit the possibility of an all-singleton configuration, we let $q = N$. This representation was implemented by Gebbie et al. in their serial GA and was adopted in this research [101].

6.8.3.4 Fitness function

The Giada and Marsili maximum log-likelihood function $\mathcal{L}_c$, as shown in Equation 6.10, was used as the fitness function. This is used to determine whether the cluster configuration represents the inherent structure of the data set, i.e. it will be used to detect if the GA converges to the fittest individual, which will represent a cluster configuration of correlated assets or objects in the data set.

6.8.3.5 Master-slave PGA implementation

The unparalellised MATLAB GA implementation of the likelihood function by Gebbie, Wilcox and Mbambiso [101] served as a starting point. In order to maximise the performance of the GA, the application of genetic operators and evaluation of the fitness function were parallelised for the CUDA framework [55]. A summarised exposition is presented here.

Emphasis was placed on outsourcing as much of the GA execution to the GPU and made use of GPU memory as extensively as possible [256]. The master-slave PGA uses a single population, where evaluation of the individuals and successive application of genetic operators are conducted in parallel. The global parallelisation model does not predicate anything about the underlying computer architecture, so it can be implemented efficiently on a shared-memory and distributed-memory model platform [223]. By delegating these tasks to the GPU and making extensive use of GPU memory, this minimises the data transfers between the host and device. These transfers have a significantly lower bandwidth than data transfers between shared or global memory and the kernel executing on the GPU. The algorithm in [101] was modified to maximise the

performance of the master-slave PGA and have a clear distinction between the master node (CPU), which controls the evolutionary process by issuing the commands for the GA operations to be performed by the slave nodes (GPU streaming multiprocessors). The pseudo-code for the algorithm implemented is shown in Algorithm 2.

Algorithm 2 Master-slave PGA for cluster identification

```
Initialise ecosystem for evolution
Size the thread blocks and grid to achieve greatest parallelisation
ON GPU: Create initial population
while TRUE do
    ON GPU: Evaluate fitness of all individuals
    ON GPU: Evaluate state and statistics
    ON GPU: Determine if termination criteria are met
    if YES then
        Terminate ALGO; Exit While loop;
    else
        Continue
    end if
    ON GPU: Isolate fittest individuals
    ON GPU: Apply elitism
    ON GPU: Apply scaling
    ON GPU: Apply genetic operator: selection
    ON GPU: Apply genetic operator: crossover
    ON GPU: Apply genetic operator: mutation
    ON GPU: Apply replacement (new generation created)
end while
Report on results
Clean-up (Deallocate memory on GPU/CPU; Release device)
```

To achieve data parallelism and make use of the CUDA thread hierarchy, we mapped individual genes onto a 2-dimensional grid. Using the representation shown in Equation 6.11, assuming a population of 400 individuals and 18 stocks:

$$\mathit{Individual}_1 = \{1, 2, 4, 5, 7, ..., 6\}$$

$$\mathit{Individual}_2 = \{9, 2, 1, 1, 1, ..., 2\}$$

$$\mathit{Individual}_3 = \{3, 1, 3, 4, 6, ..., 2\}$$

$$\vdots$$

$$\mathit{Individual}_{400} = \{8, 1, 9, 8, 7, ..., 3\}$$

would be mapped to grid cells, as illustrated in Figure 6.2. The data grid cells are mapped to threads, where each thread executes a kernel processing the data cell at the respective xy-coordinate.

	Individual 1	Individual 2	Individual 3	Individual 4	Individual 5	...	Individual 400
	GRID						
	Block (0,0)	*Block (1,0)*	*Block (2,0)*	*Block (3,0)*	*Block (4,0)*	...	*Block (399,0)*
Stock 1	1	9	3	2	5		8
	Block (0,1)	*Block (1,1)*	*Block (2,1)*	*Block (3,1)*	*Block (4,1)*	...	*Block (399,1)*
Stock 2	2	2	1	1	7		1
	Block (0,2)	*Block (1,2)*	*Block (2,2)*	*Block (3,2)*	*Block (4,2)*	...	*Block (399,2)*
Stock 3	4	1	3	3	3		9
	Block (0,3)	*Block (1,3)*	*Block (2,3)*	*Block (3,3)*	*Block (4,3)*	...	*Block (399,3)*
Stock 4	5	1	4	3	4		8
	Block (0,4)	*Block (1,4)*	*Block (2,4)*	*Block (3,4)*	*Block (4,4)*	...	*Block (399,4)*
Stock 5	7	1	6	7	8		7
⋮	⋮	⋮	⋮	⋮	⋮		⋮
	Block (0,17)	*Block (1,17)*	*Block (2,17)*	*Block (3,17)*	*Block (4,17)*	...	*Block (399,17)*
Stock 18	6	2	2	1	2		3

FIGURE 6.2: Mapping of individuals onto the CUDA thread hierarchy

Graphics card	**Nvidia GTX Titan Black**	**Nvidia Tesla C2050**
Compute capability	3.5	2.0
SMs	15	14
Max threads / thread block	1024	1024
Thread block dimension	32	32
Max thread blocks / multiprocessor	16	8
Max number of stocks	3840	3584
Max population size	17 472	18 720

TABLE 6.2: Restrictions on number of stocks and population size. For the Tesla card, *Max number of stocks* $= (14) * (1024/32) * 8 = 3584$ and *Max population size* $= (65535/(3584/32)) * 32 = 18720$.

Given the hardware used in this investigation (see Table 6.1), Table 6.2 outlines the restrictions on the permissible stock universe and population sizes imposed by the chosen mapping of individual genes to threads. A thread block dimension of 32 is chosen for larger problems, since this ensures that the permissible population size is larger than the number of stocks to cluster.

We note that the efficiency of the algorithm may be compromised near the physical limits outlined in Table 6.2, since the CUDA memory hierarchy would force threads to access high-latency global memory banks more often. However, for the particular domain problem we are considering here, the Johannesburg Stock Exchange consists of around 400 listed companies on its main board, which represents an upper limit on the number of stocks of interest for local cluster analysis. This is well within the physical limits of the algorithm, while still providing scope to extend the application to multiple markets. For applications with a large number of objects, one could make use of Nvidia's Scalable Link Interface (SLI) technology to link a number of physical graphics cards and

pool their memory and processing resources [201]. The CUDA implementation would recognise the linked cards as a single card with increased memory and thread capacity, and the computation would scale accordingly.

The details on the full implementation, as well as specific choices regarding initialisation, block sizes and threads per block, are given in [55].

6.8.3.6 Key implementation challenges

A key challenge in CUDA programming is adapting to the Single Program Multiple Data (SPMD) paradigm, where multiple instances of a single program use unique offsets to manipulate portions of a block of data [65]. This architecture suits data parallelism, whereas task parallelism requires a special effort. In addition, since each warp (group of 32 threads) is executed on a single SPMD processor, divergent threads in a warp can severely impact performance. In order to exploit all processing elements in the multiprocessor, a single instruction is used to process data from each thread. However, if one thread needs to execute different instructions due to a conditional divergence, all other threads must effectively wait until the divergent thread re-joins them. Thus, divergence forces sequential thread execution, negating a large benefit provided by SPMD processing.

The CUDA memory hierarchy contains numerous shared memory banks which act as a common data cache for threads in a thread block. In order to achieve full throughput, each thread must access a distinct bank and avoid bank conflicts, which would result in additional memory requests and reduce efficiency. In our implementation, bank conflicts were avoided by using padding, where shared memory is padded with an extra element such that neighbouring elements are stored in different banks [49].

CUDA provides a simple and efficient mechanism for thread synchronisation within a thread block via the *__syncthreads()* barrier function, however inter-block communication is not directly supported during the execution of a kernel. Given that the genetic operators can only be applied once the entire population fitness is calculated, it is necessary to synchronise thread blocks assigned to the fitness computation operation. We implemented the CPU implicit synchronisation scheme [203, 252]. Since kernel launches are asynchronous, successive kernel launches are pipelined and thus the executions are implicitly synchronised with the previous launch, with the exception of the first kernel launch. Given the latency incurred on calls between the CPU and GPU, and the consequent drag on performance, GPU synchronisation schemes were explored which achieve the required inter-block communication. In particular, GPU simple synchronisation,

GPU tree-based synchronisation and GPU lock-free synchronisation were considered [252].

Ultimately, the GPU synchronisation schemes were too restrictive for our particular problem, since the number of thread blocks would have an upper bound equal to the number of SMs on the GPU card. If the number of thread blocks is larger than the number of SMs on the card, execution may deadlock. This could be caused by the warp scheduling behaviour of the GPU, whereby active thread blocks resident on a SM may remain in a *busy waiting state*, waiting for unscheduled thread blocks to reach the synchronisation point. While this scheme may be more efficient for smaller problems, we chose the CPU synchronisation scheme in the interest of relative scalability.

6.9 Results

6.9.1 Data description

The data for this study constituted tick-level trades and top-of-book quotes for 42 stocks on the Johannesburg Stock Exchange (JSE) from 1 November 2012 to 30 November 2012. This data was sourced from the Thomson Reuters Tick History (TRTH) database. The raw data was aggregated according to the time-scale considered (5-minute, 15-minute, 30-minute and 60-minute), before calculating the required features (change in trade price, trade volume, spread and volume imbalance). The 42 stocks considered represent the prevailing constituents of the FTSE/JSE Top40 headline index, which contains the 42 largest stocks by market capitalisation in the main board's FTSE/JSE All-Share index.

The objects of interest for the cluster analysis are the *time periods*. Table 6.3 provides an example of the required data returns matrix, from which a correlation matrix is computed for time period similarity. This is the only required input for the clustering algorithm.

	Feature	Times					
		01-Nov-2012 09:00	01-Nov-2012 09:15	01-Nov-2012 09:30	...	30-Nov-2012 16:30	30-Nov-2012 16:45
Trade Price	AGL trade price return	0.35	0.60	0.85	...	0.39	0.22
	AMS trade price return	0.94	0.71	0.73	...	0.63	0.78
	SBK trade price return	0.70	0.38	0.58	...	0.38	0.81
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	WHL trade price return	0.90	0.49	0.05	...	0.65	0.53
Spread	AGL spread return	0.64	0.49	0.68	...	0.05	0.95
	AMS spread return	0.33	0.09	0.76	...	0.44	0.97
	SBK spread return	0.09	0.73	0.54	...	0.80	0.48
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	WHL spread return	0.41	0.61	0.11	...	0.40	0.69
Trade Volume	AGL trade volume return	0.61	0.59	0.96	...	0.65	0.50
	AMS trade volume return	0.16	0.09	0.47	...	0.86	0.57
	SBK trade volume return	0.98	0.05	0.67	...	0.72	0.12
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	WHL trade volume return	0.38	0.49	0.36	...	0.27	0.81
Volume Imbalance	AGL volume imb return	0.01	0.45	0.78	...	0.69	0.77
	AMS volume imb return	0.54	0.17	0.87	...	0.47	0.44
	SBK volume imb return	0.20	0.42	0.91	...	0.88	0.58
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	WHL volume imb return	0.20	0.09	0.38	...	0.90	0.12

TABLE 6.3: Illustration of data returns matrix as an input for estimation of 15-minute period correlations

6.9.2 Workflow

Figure 6.3 illustrates the process workflow and tools used for performing the temporal cluster analysis. The TRTH tick data is stored in a MongoDB noSQL database, with optimised query indexes for efficient data retrieval. A bespoke Application Programming Interface (API) was written to transport data from MongoDB to our primary scientific computing platform, MATLAB. The data is used to instantiate a High Frequency Time Series (HFTS) object in MATLAB, which allows for efficient merging, resampling and aggregation of large-scale irregularly-spaced tick data. Based on a chosen time-scale, the data is aggregated, features are extracted and returns calculated, before computing the time period correlation matrix. The PGA was implemented in CUDA-C using Nvidia Nsight and the Microsoft Visual Studio development environment. The compiled PGA was called from the MATLAB environment to run the temporal cluster analysis. The resulting cluster configuration is transported to the MATLAB workspace, from which we can determine the power-law fits, extract SSVs, estimate online clusters and compute transition probability matrices. Using the stock, time period, cluster configuration and correlation data, a MATLAB script was written to generate an XML file containing the required node and edge metadata for an undirected graph to import into Gephi. Gephi was used for cluster configuration visualisation, as described in Section 6.9.3.

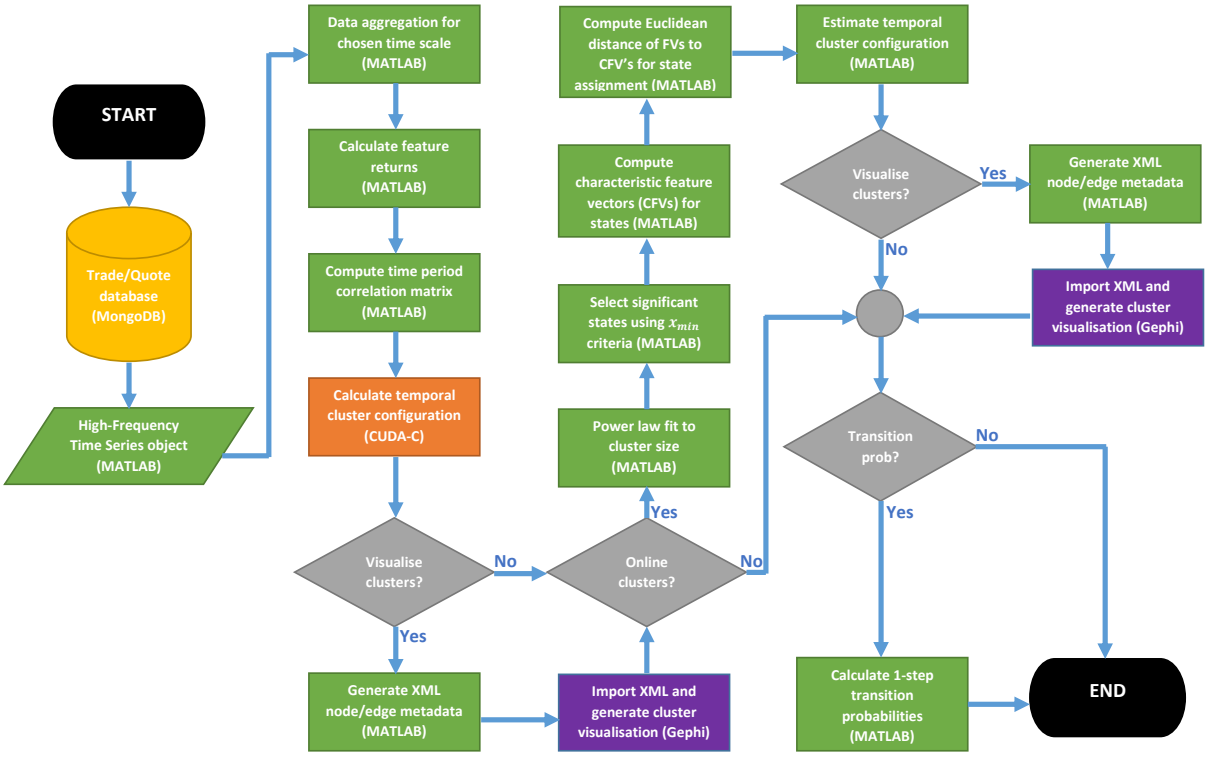


FIGURE 6.3: Flowchart illustrating workflow to determine the temporal cluster configuration from a time period correlation matrix, identify persistent states, estimate temporal cluster configuration using feature vectors and determine state transition probabilities. Processes are coloured by platform: MongoDB = Yellow, MATLAB = Green, CUDA-C = Orange, Gephi = Purple.

6.9.3 Visualisation

For the cluster configuration visualisation, we made use of the Gephi graph visualisation and manipulation software package [27], with a customised enumeration of nodes and edges and the Fruchterman-Reingold [93] node spacing algorithm. The presence of an edge between nodes indicates membership to the same cluster, while edge thickness provides a visual impression of object-object correlation, and hence intra-cluster connectedness. For the visualisations which follow, we chose to colour the nodes by intraday time period, in order to illuminate any calendar time effects in the detected states. According to this scheme, the same time on different days will receive the same colour. These visualisations are shown in Figures 6.4, 6.6, 6.8, 6.10, 6.13, 6.14, 6.15 and 6.16.

6.9.4 Results discussion

For each set of results, we consider 8 hours of continuous trading activity each day, from 09:00 to 17:00, for the duration of one month. Figure 6.4 shows the temporal cluster configuration of 60-minute periods. We first note that the detection of non-trivial clusters from microstructure-based time correlations indicates that intraday dynamics may be reducible to a finite set of temporal states. Considering the time-of-day colour shading, we notice two clusters which exhibit market activity characteristics which coincide with morning and afternoon times. The dark green cluster refers to the first hour of the trading day (09:00 to 10:00), which incorporates opening auction and subsequent activity. We note that the South African equity market is strongly influenced by global market activity, in part due to local stocks being listed on multiple exchanges in the UK, USA, Europe and Australia [145]. During the period considered in this analysis, the UK market open occurred at 10:00 SAST and US market open at 15:30 SAST. The UK market open has a significant impact on local trading dynamics, with the 10:00 to 11:00 periods dispersing across clusters with no discernible time-of-day correlation. We note a contiguous dark orange cluster emerge from 15:00 to 16:00, as the US market starts to participate in local trading activity. This pattern of market activity broadly corroborates these exogenous market effects from global markets. Figure 6.5 shows the SSVs extracted from the significant states selected from Figure 6.4. As discussed in Section 6.7, we used the x_{min} statistic from the power-law fit to the tail distribution of cluster sizes to determine the significant states. For the 60-minute periods, the most significant power-law fit was for cluster sizes ≥ 13, resulting in 6 significant states. The resulting SSVs are all relatively different, when considering the magnitude and direction of each of the average change in feature values. This ensures greater certainty in the state assignment of an online FV.

Figure 6.6 shows the temporal cluster configuration of 30-minute periods. We see a larger number of states emerge as the granularity increases, with 60-minute states being dissected based on finer-grained market activity. The dark green and dark orange contiguous morning and afternoon states still persist at this scale, although endogenous system characteristics begin to mask previously identified exogenous characteristics. We note that there is no defined hierarchy emerging, in that a set of 30-minute clusters cannot be combined to form the 60-minute clusters identified previously, further highlighting time-scale-specific behaviour. Figure 6.7 shows the SSVs of significant states, based on the 10 clusters with a size ≥ 14.

Figure 6.8 shows the temporal cluster configuration of 15-minute periods. We notice increasing time-of-day diversity in each of the identified clusters, further highlighting endogenous system activity. The red contiguous cluster is associated with the period

from 16:30 to 16:45, suggesting a particular signature of market activity leading into the closing auction, which starts at 16:50. The UK and US related effects seem to have a weaker impact at this scale, with exchange-specific rules having a more dominant effect. As a result, we see a larger variety of SSVs in Figure 6.9, some with similar profiles seen at the 30-minute scale, but with a larger focus on magnitude, rather than merely direction.

Figure 6.10 shows the temporal cluster configuration of 5-minute periods. Here we see quite a different profile of system behaviour. There are a large number of singletons, which could be attributed to the amount of noise in the data at this scale, making it more difficult to discern significant structure. We notice an interesting time-of-day correlation with detected clusters, however broad periods (morning, lunch, afternoon) appear to have been dissected into contiguous blocks based on state-specific market activity. The 5-minute time scale is starting to capture the effects of automated, rule-based trading agents which shows quite a different characteristic signature. This further highlights the importance of studying market activity profiles at the scale at which you intend to participate. Even when one considers the associated SSVs in Figures 6.11 and 6.9, the 5-minute and 15-minute studies exhibit the same number of significant states using the power-law criterion, however the combinations of direction and magnitude for the feature values are quite different.

Figure 6.12 illustrates the results of the power-law fits to the cluster size empirical distribution at each time scale. Each fit to the tail distribution exhibits a Kolmogorov-Smirnov p-value > 0.1 (assuming a null hypothesis of a power-law fit), suggesting a strong fit of the power-law functional form for the given scaling factor (α) and minimum size (x_{min}) [58]. In addition, we note the α exponents are different for each of the time scales considered, suggesting different universality classes of system behaviour at different time scales. This behaviour is interesting and needs to be verified in a more comprehensive study with larger datasets and a stability analysis, however these preliminary results do indicate the presence of some hierarchy of system behaviour, motivating the need for scale-specific temporal analysis.

Figure 6.13 shows the estimated 60-minute cluster configuration for the same period (1 November 2012 to 30 November 2012), but where the distance of each period's FV to the identified SSVs is used as the criterion for state assignment. This is a simple in-sample test to determine whether the proposed scheme for online state assignment can discern the structure suggested by direct application of the clustering algorithm. By comparing Figure 6.13 and Figure 6.4, we notice that the online state assignment algorithm does recover the contiguous morning and afternoon states, but more broadly intuitively separates periods into: opening auction and early morning trading state, UK

market open state, two lunch states, US market open state and a end-of-day/closing auction state. This completely captures the exogenous market effects, which is a strong validation for the approach. Table 6.5 shows an empirical 1-step transition probability matrix calculated from the states shown in Figure 6.13, illustrating one potential application of this technique. The 1-step transitions show a particular preference, suggesting some predictability which can be exploited by trading agents. To be clear, the online assignment of a FV to a state means that we have developed a mechanism to *detect* which state we are currently in, using the prevailing set of SSVs. The transition matrix can be used and updated online, and for optimal planning in the domain.

Figures 6.14 to 6.16 and Tables 6.6 to 6.8 show the estimated cluster configurations and transition probability matrices using the SSVs at the specified time scale. It is interesting to observe the dilution of the exogenous time-of-day effects as one approaches the 5-minute scale.

Figure 6.17 illustrates the stability of the online state assignment algorithm out-of-sample. Given that the state assignment of an online FV is based on the minimum Euclidean distance to predetermined SSVs, we compute the *best match* distance for each of the FVs in a sample and use a boxplot to visualise the empirical distribution. Here we propose offline estimation of SSVs used for online state detection. The online cluster configurations shown in Figures 6.13, 6.14, 6.15 and 6.16 use FVs from the *ex-ante* period, i.e. the same period used to estimate the SSVs. It is prudent to determine whether state assignment using out-of-sample (*ex-post*) FVs deviate significantly from in-sample assignment, and gauge the out-of-sample efficacy of the SSVs before re-estimation is necessary. Given the computation times shown in Table 6.4, in practice one could estimate the SSVs overnight for each trading day. We have considered SSVs estimated from the period 1 November 2012 to 30 November 2012, and compared the resulting online states from the *ex-ante* period (1 November 2012 to 30 November 2012) with states from an *ex-post* period (3 December 2012 to 7 December 2012, one week after SSV estimation). From these results, it appears that 60-minute states cannot be reliably determined *ex-post* using the online detection algorithm, given the observed higher range of *best match* Euclidean distances. The 30-minute, 15-minute and 5-minute time scales all exhibit acceptable *ex-post best match* distances, with the exception of a few outliers. From these preliminary results, it appears that the algorithm can be used to reliably determine 30-minute, 15-minute and 5-minute states for a relatively short *ex-post* period following SSV estimation. A more robust study should consider the precise half-life of the SSVs, but given the relatively fast computation time, this is unlikely to be a practical concern.

The average computation times indicated in Table 6.4 are not overly onerous, suggesting that for practical application, overnight or even intraday estimation of cluster configurations to capture recent dynamics is feasible. The proposed PGA thus offers an efficient, scalable alternative for finding the best approximation of the optimal cluster configuration, suitable for clustering objects on multiple observable features. We note that the number of generations and stall generations indicated in Table 6.4 are higher than one would typically specify for a genetic algorithm, since these promote potential over-fitting to the prescribed dataset. Recall that our application is to find the candidate cluster configuration which best explains the structure inherent in a given correlation matrix. Thus we are not concerned with out-of-sample validity, but would rather prefer to find a configuration with the highest likelihood value. The higher number of generations and stall generations, together with the mutation operator, promotes convergence to a higher likelihood structure.

Time scale	Number of periods (objects)	Population size	Generations	Stall generations	Mutation probability	Crossover probability	Computation Time (sec)$*$
5-minute	2208	4000	4000	1000	0.09	0.9	603 (D)
15-minute	736	1000	4000	500	0.09	0.9	382 (N)
30-minute	368	800	4000	500	0.09	0.9	215 (N)
60-minute	184	600	4000	500	0.09	0.9	132 (N)

TABLE 6.4: Parameter values and computation times for Parallel Genetic Algorithm
$*$ Average from 20 independent runs; N refers to the GTX765m Notebook GPU and D refers to the GTX Titan X Desktop GPU.

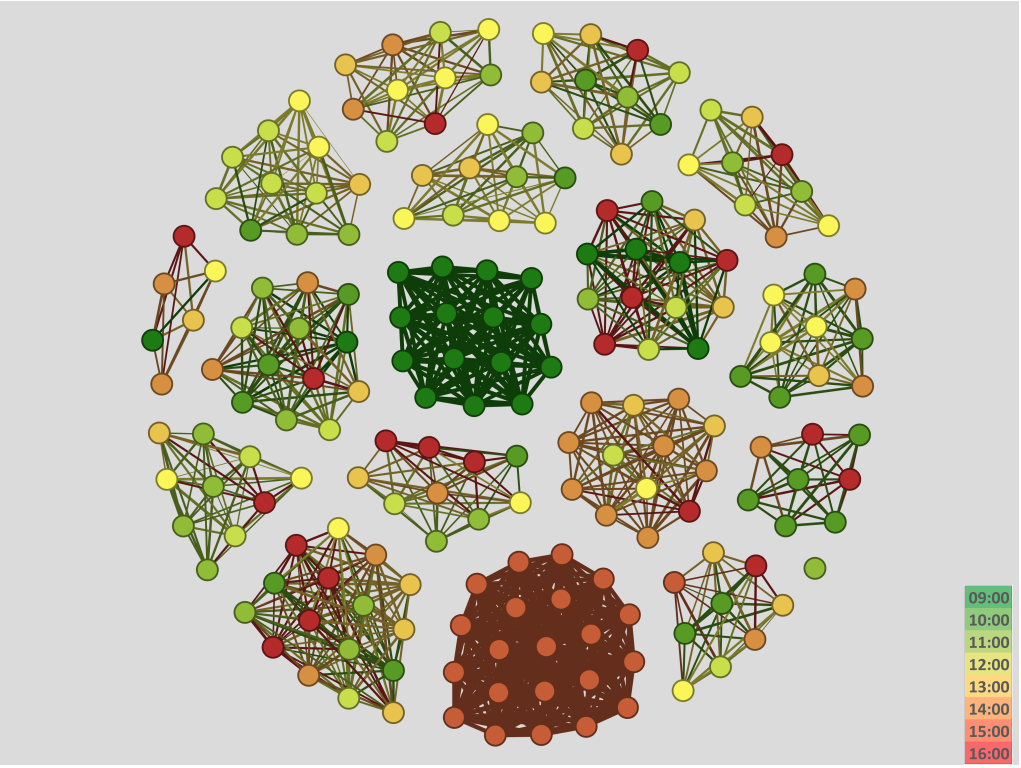


FIGURE 6.4: JSE TOP40 60-minute temporal clusters for the period 01-Nov-2012 to 30-Nov-2012, representing 184 distinct periods. Each node represents a 60-minute period during a trading day, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership.

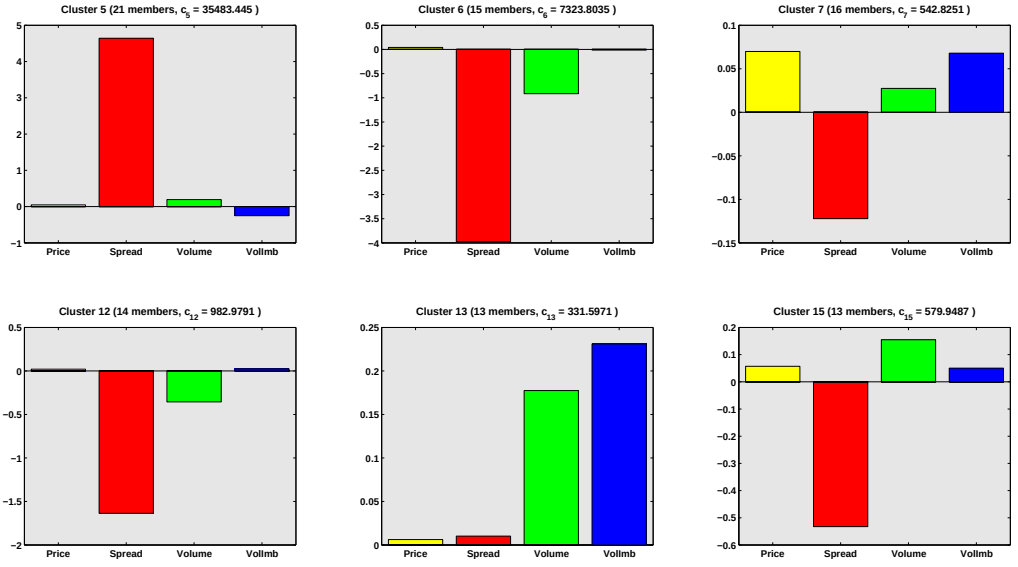


FIGURE 6.5: JSE TOP40 60-minute cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses.

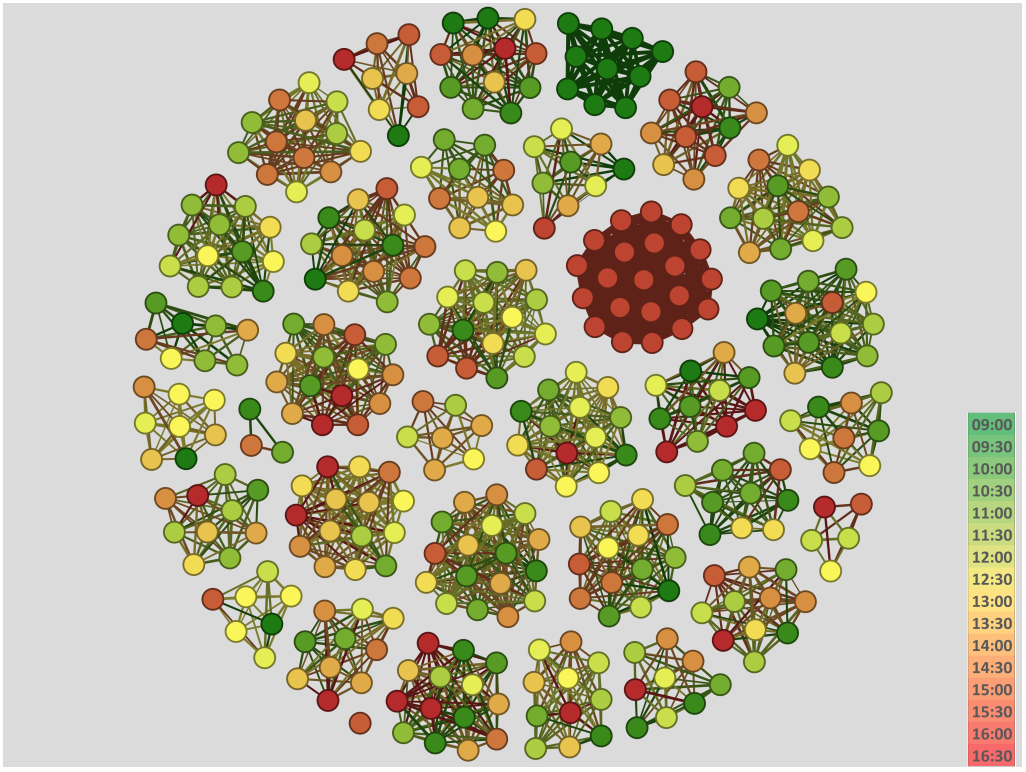


FIGURE 6.6: JSE TOP40 30-minute temporal clusters for the period 01-Nov-2012 to 30-Nov-2012, representing 368 distinct periods. Each node represents a 30-minute period during a trading day, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership.

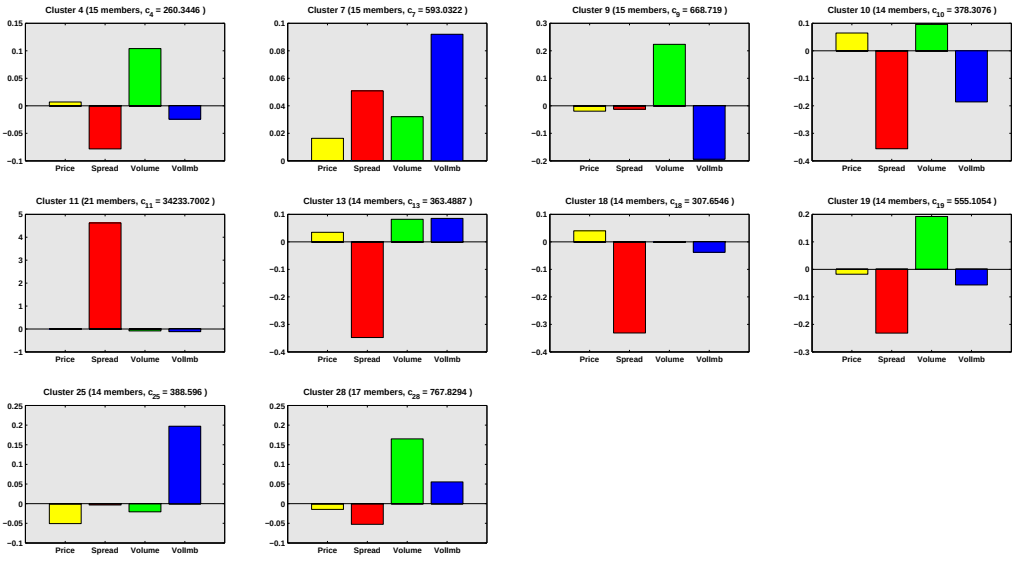


FIGURE 6.7: JSE TOP40 30-minute cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses.

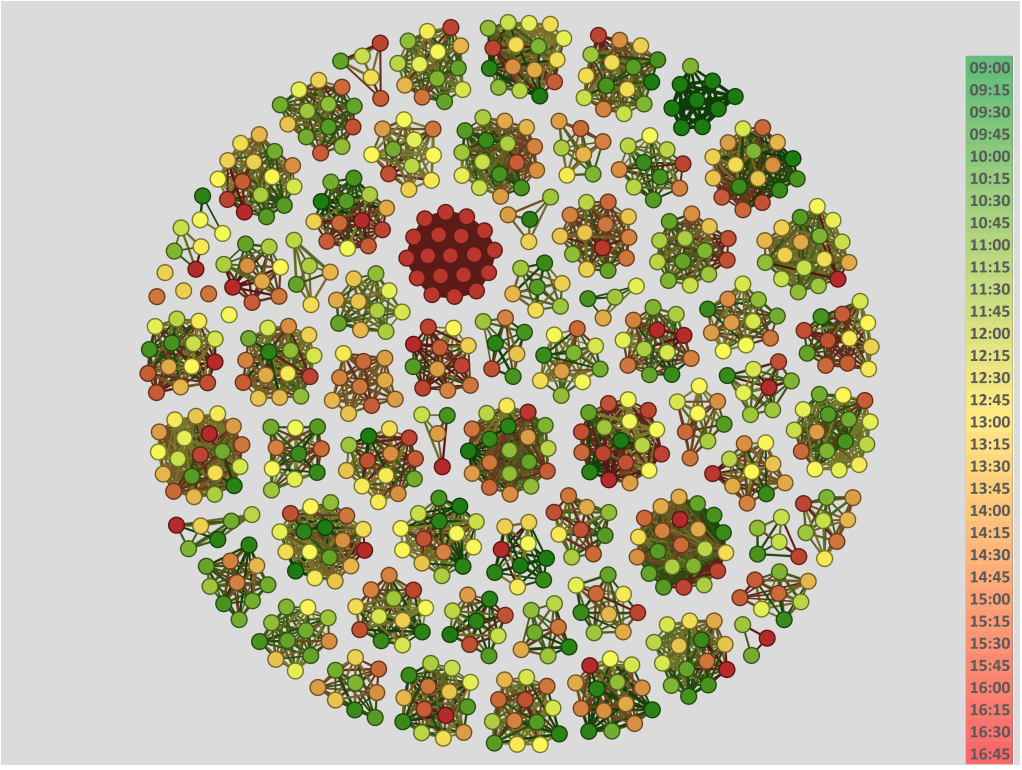


FIGURE 6.8: JSE TOP40 15-minute temporal clusters for the period 01-Nov-2012 to 30-Nov-2012, representing 736 distinct periods. Each node represents a 15-minute period during a trading day, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership.

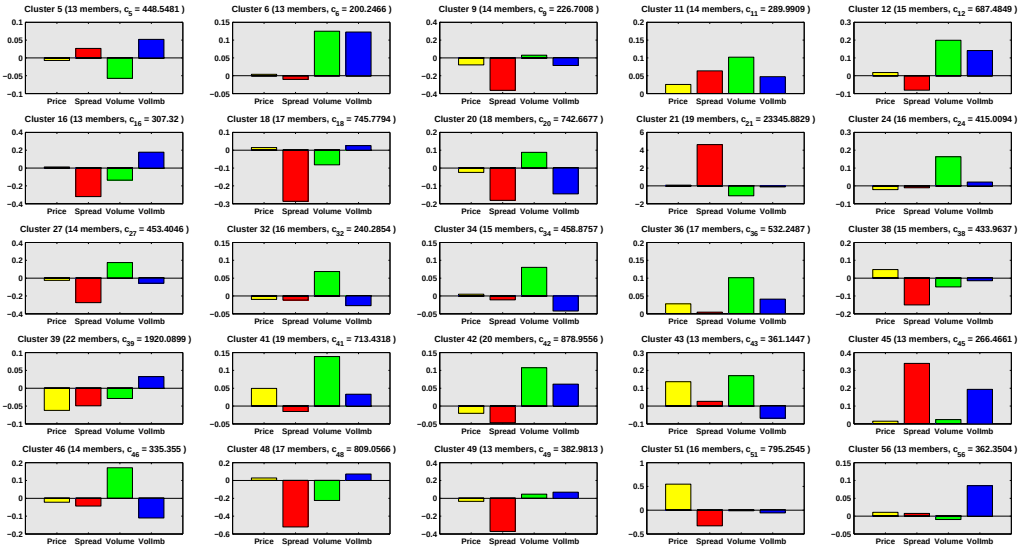


FIGURE 6.9: JSE TOP40 15-minute cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses.

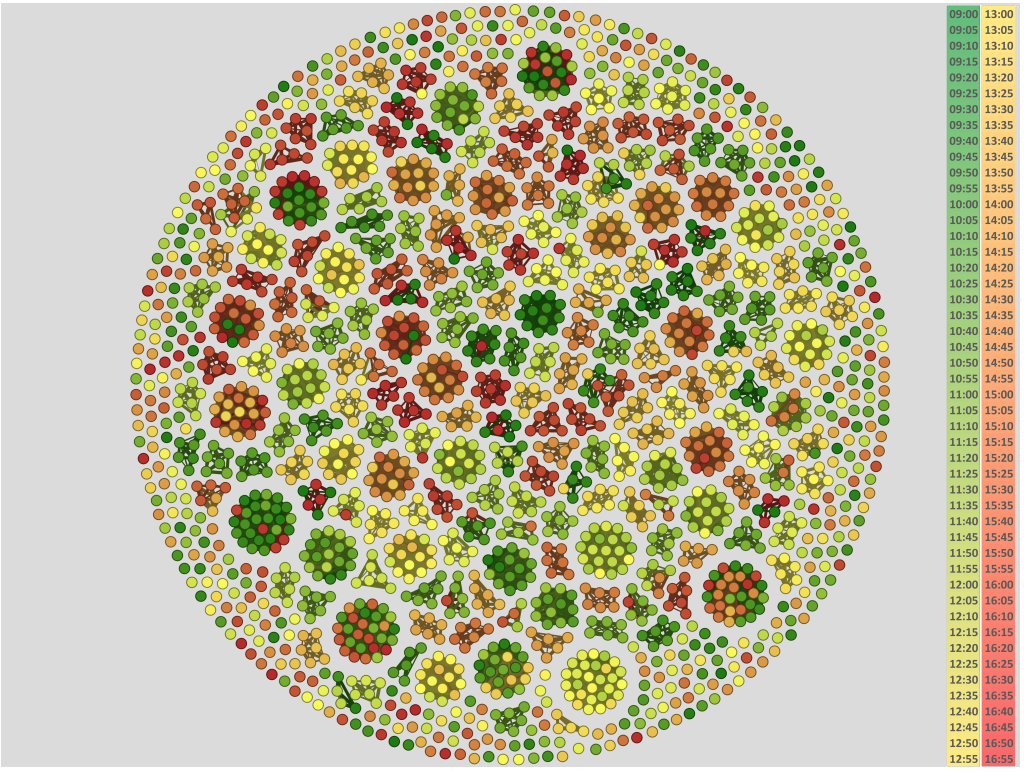


FIGURE 6.10: JSE TOP40 5-minute temporal clusters for the period 01-Nov-2012 to 30-Nov-2012, representing 2208 distinct periods. Each node represents a 5-minute period during a trading day, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership.

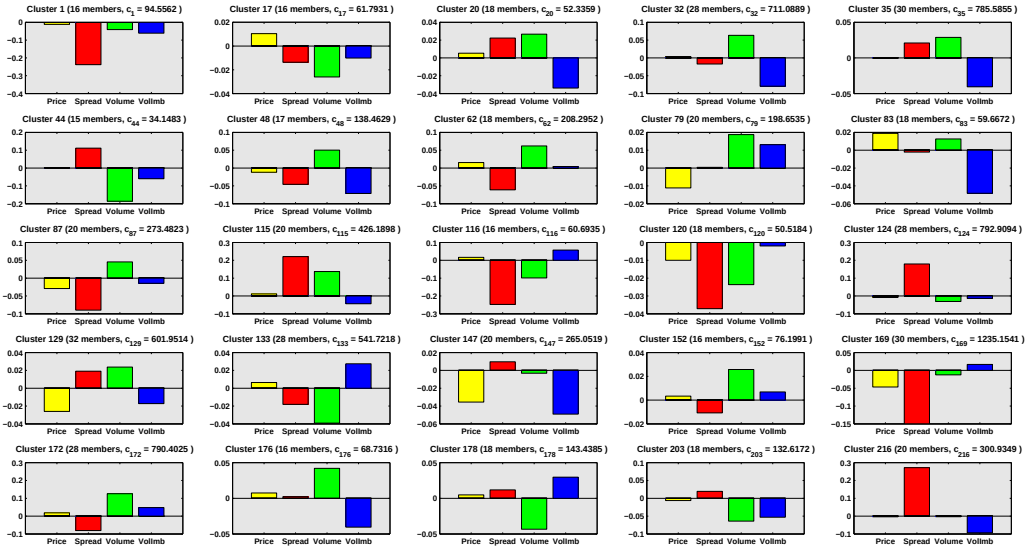
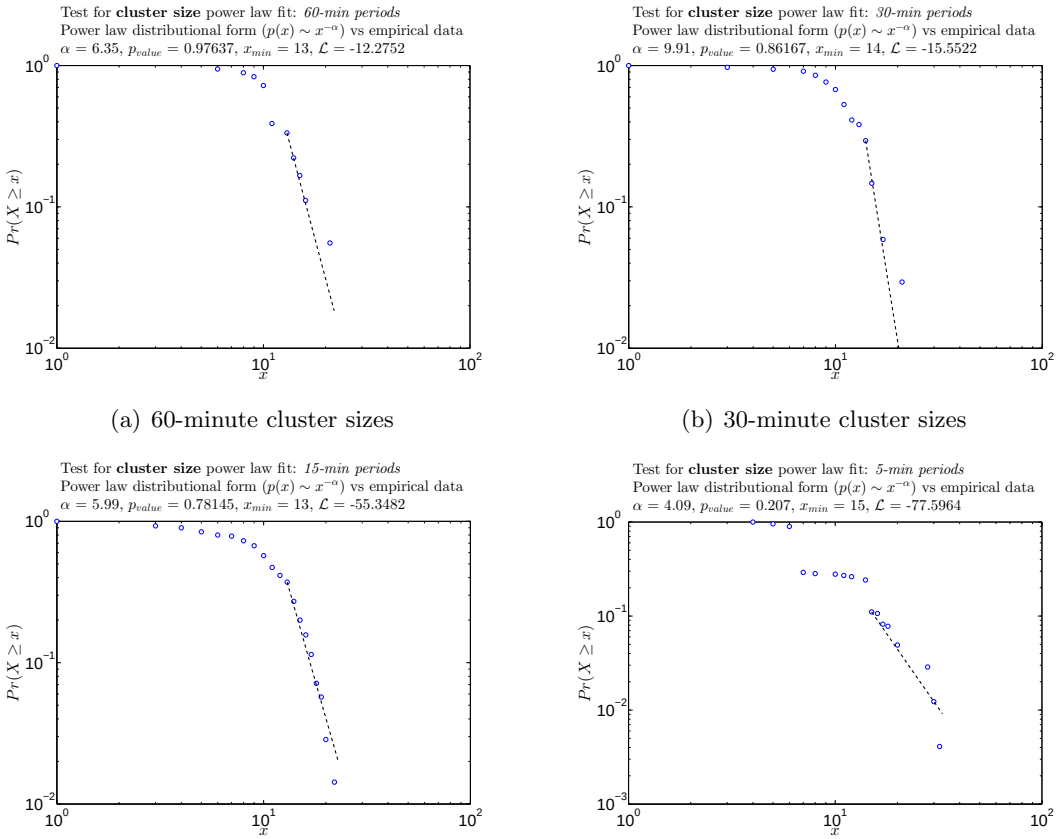


FIGURE 6.11: JSE TOP40 5-minute cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses.



(a) 60-minute cluster sizes

(b) 30-minute cluster sizes

(c) 15-minute cluster sizes

(d) 5-minute cluster sizes

FIGURE 6.12: Testing conjecture of power law fit for varying time scale cluster sizes, applying the Clauset, Shalizi and Newman algorithm [58]. α indicates the scaling parameter of the proposed fit, p_{value} indicates the p-value from a Kolmogorov-Smirnov test for the goodness-of-fit of the proposed model to the data, x_{min} indicates the lower-bound for the power law fit and $\mathcal{L}$ is the log-likelihood of the data ($x \geq x_{min}$) under the power law fit.

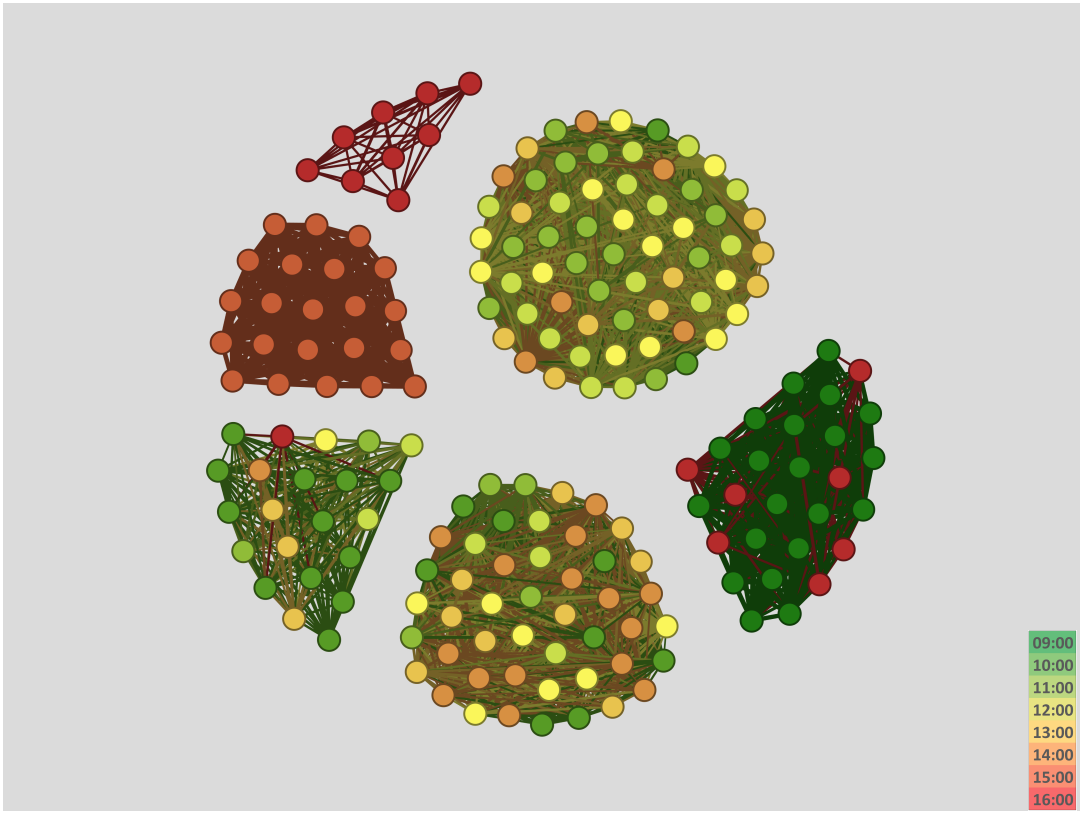


FIGURE 6.13: Estimated 60-minute clusters using identified state signature vectors. The Euclidean distance is calculated between each temporal period's feature vector and the state signature vectors. Cluster index assignment is based on the state signature vector which yields the minimum distance.

		$\mathbf{state_{t+1}}$					
		1	**2**	**3**	**4**	**5**	**6**
$\mathbf{state_t}$	**1**	0.13	0.49	0.32	0.00	0.06	0.00
	2	0.41	0.41	0.09	0.00	0.09	0.00
	3	0.00	0.00	0.00	0.52	0.05	0.43
	4	0.25	0.07	0.00	0.25	0.43	0.00
	5	0.32	0.59	0.05	0.05	0.00	0.00
	6	0.00	0.00	0.00	1.00	0.00	0.00

TABLE 6.5: Empirical 1-step transition probability matrix for 60-minute states, based on identified temporal cluster configuration. State transitions with a probability > 0 are highlighted in green.

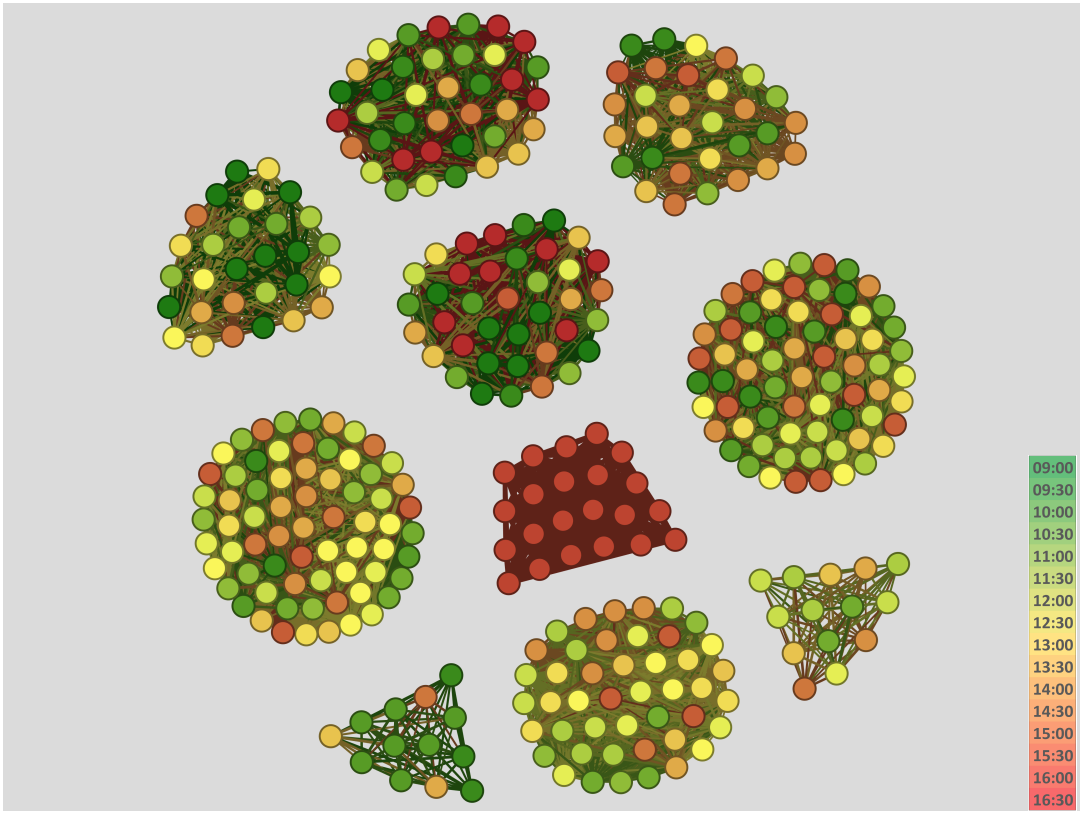


FIGURE 6.14: Estimated 30-minute clusters using identified state signature vectors. The Euclidean distance is calculated between each temporal period's feature vector and the state signature vectors. Cluster index assignment is based on the state signature vector which yields the minimum distance.

		state_{t+1}									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
state_t	**1**	0.11	0.37	0.03	0.16	0.18	0.05	0.03	0.03	0.03	0.03
	2	0.07	0.04	0.35	0.06	0.04	0.10	0.17	0.10	0.04	0.01
	3	0.06	0.33	0.10	0.07	0.17	0.09	0.06	0.04	0.03	0.04
	4	0.05	0.08	0.26	0.03	0.21	0.18	0.00	0.13	0.03	0.03
	5	0.07	0.13	0.24	0.11	0.04	0.09	0.07	0.13	0.04	0.07
	6	0.10	0.21	0.10	0.03	0.14	0.03	0.00	0.17	0.10	0.10
	7	0.57	0.00	0.00	0.38	0.00	0.05	0.00	0.00	0.00	0.00
	8	0.13	0.23	0.16	0.16	0.13	0.03	0.06	0.06	0.03	0.00
	9	0.00	0.15	0.38	0.15	0.08	0.00	0.00	0.08	0.00	0.15
	10	0.00	0.36	0.21	0.07	0.29	0.00	0.00	0.07	0.00	0.00

TABLE 6.6: Empirical 1-step transition probability matrix for 30-minute states, based on identified temporal cluster configuration. State transitions with a probability > 0 are highlighted in green.

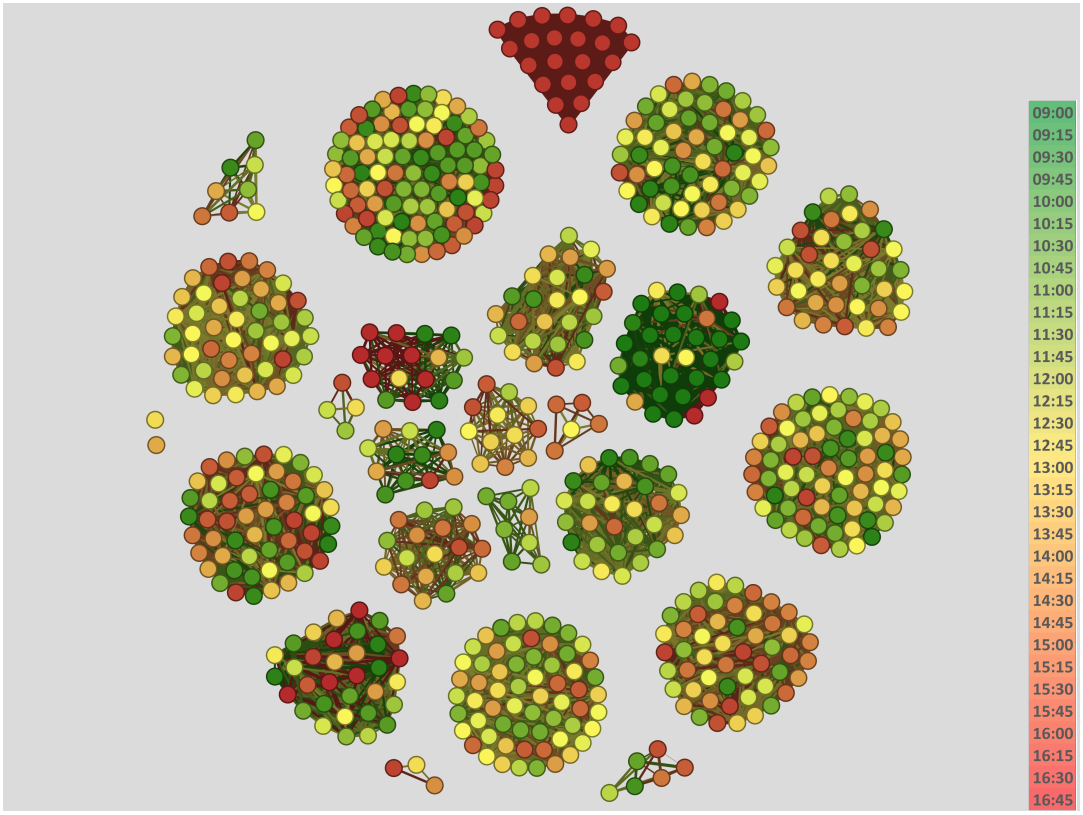


FIGURE 6.15: Estimated 15-minute clusters using identified state signature vectors. The Euclidean distance is calculated between each temporal period's feature vector and the state signature vectors. Cluster index assignment is based on the state signature vector which yields the minimum distance.

	state_{t+1}																								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
1	0.00	0.20	0.13	0.00	0.07	0.00	0.03	0.17	0.10	0.07	0.03	0.03	0.03	0.03	0.00	0.00	0.00	0.03	0.07	0.00	0.00	0.00	0.00	0.00	0.00
2	0.07	0.03	0.07	0.08	0.09	0.01	0.01	0.08	0.03	0.07	0.02	0.04	0.02	0.03	0.18	0.08	0.02	0.00	0.01	0.01	0.02	0.00	0.00	0.00	0.00
3	0.02	0.12	0.02	0.20	0.13	0.00	0.00	0.03	0.03	0.17	0.03	0.05	0.00	0.00	0.17	0.00	0.00	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00
4	0.07	0.16	0.02	0.07	0.09	0.00	0.02	0.13	0.00	0.04	0.00	0.09	0.05	0.00	0.02	0.07	0.00	0.00	0.07	0.05	0.00	0.02	0.02	0.00	0.00
5	0.03	0.28	0.15	0.08	0.00	0.03	0.00	0.08	0.03	0.03	0.00	0.08	0.00	0.03	0.10	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00
6	0.50	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.17	0.17	0.00	0.00	0.00	0.17	0.33	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.05	0.25	0.05	0.08	0.07	0.00	0.00	0.07	0.07	0.05	0.00	0.02	0.03	0.00	0.15	0.00	0.00	0.02	0.05	0.00	0.02	0.02	0.00	0.02	0.02
9	0.00	0.12	0.06	0.03	0.00	0.00	0.00	0.12	0.00	0.09	0.03	0.24	0.00	0.00	0.06	0.03	0.18	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
10	0.04	0.10	0.19	0.00	0.02	0.00	0.00	0.19	0.02	0.04	0.00	0.02	0.04	0.02	0.12	0.06	0.06	0.00	0.02	0.02	0.02	0.02	0.00	0.00	0.00
11	0.17	0.08	0.08	0.08	0.00	0.00	0.00	0.00	0.08	0.17	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.08	0.00	0.00	0.00	0.00
state_t **12**	0.04	0.09	0.09	0.11	0.09	0.00	0.00	0.09	0.04	0.06	0.00	0.00	0.04	0.02	0.17	0.04	0.04	0.02	0.00	0.02	0.02	0.02	0.00	0.00	0.00
13	0.00	0.06	0.11	0.28	0.00	0.00	0.00	0.06	0.06	0.22	0.00	0.00	0.00	0.06	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00
14	0.00	0.42	0.08	0.00	0.00	0.00	0.08	0.08	0.00	0.08	0.00	0.17	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
15	0.01	0.18	0.15	0.01	0.01	0.01	0.01	0.07	0.03	0.04	0.01	0.21	0.07	0.00	0.03	0.03	0.00	0.00	0.07	0.03	0.00	0.00	0.00	0.00	0.00
16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.38	0.00	0.00	0.00	0.00	0.00	0.00	0.00
17	0.03	0.12	0.12	0.06	0.03	0.00	0.00	0.00	0.09	0.03	0.06	0.06	0.00	0.06	0.09	0.00	0.15	0.09	0.03	0.00	0.00	0.00	0.00	0.00	0.00
18	0.00	0.06	0.06	0.06	0.06	0.00	0.00	0.00	0.00	0.06	0.06	0.00	0.06	0.00	0.00	0.00	0.50	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00
19	0.12	0.04	0.00	0.15	0.04	0.04	0.00	0.12	0.04	0.04	0.00	0.00	0.00	0.08	0.19	0.00	0.04	0.04	0.00	0.00	0.04	0.00	0.00	0.04	0.00
20	0.00	0.25	0.13	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.00	0.13	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.00	0.00	0.00
21	0.00	0.13	0.00	0.13	0.00	0.00	0.00	0.00	0.00	0.38	0.00	0.13	0.00	0.00	0.13	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.00	0.00	0.00
22	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.20	0.20	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TABLE 6.7: Empirical 1-step transition probability matrix for 15-minute states, based on identified temporal cluster configuration. State transitions with a probability > 0 are highlighted in green.

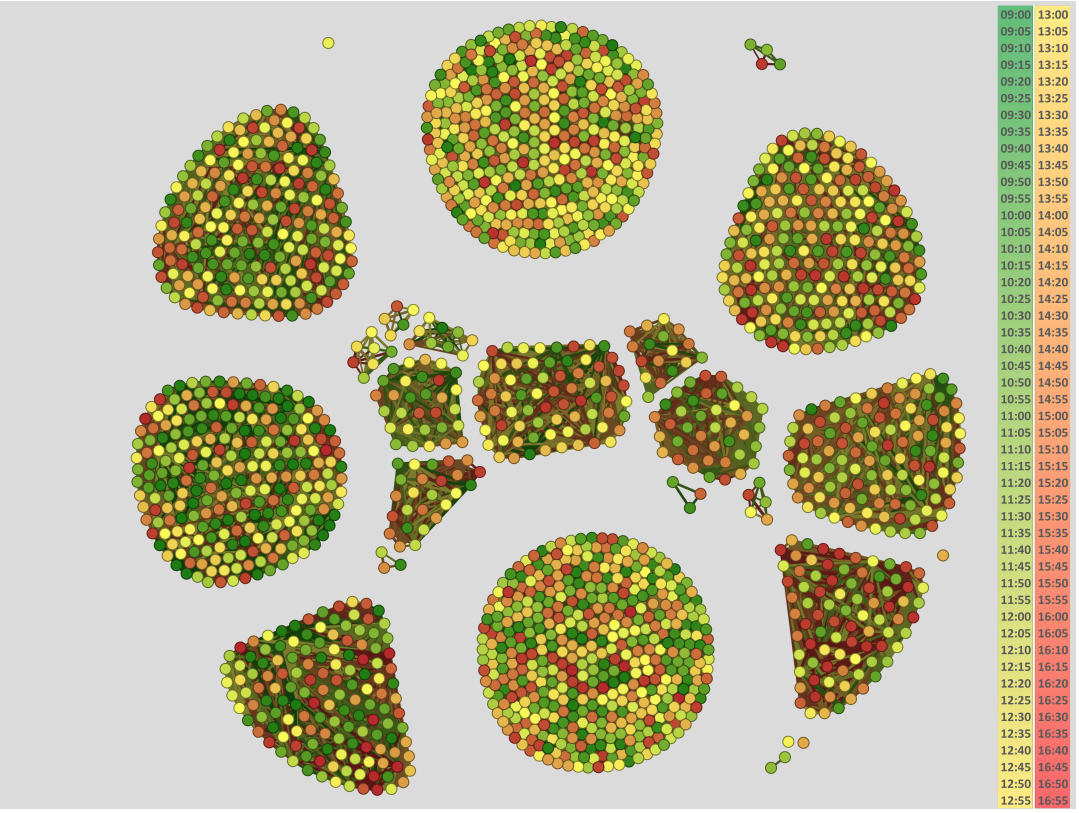
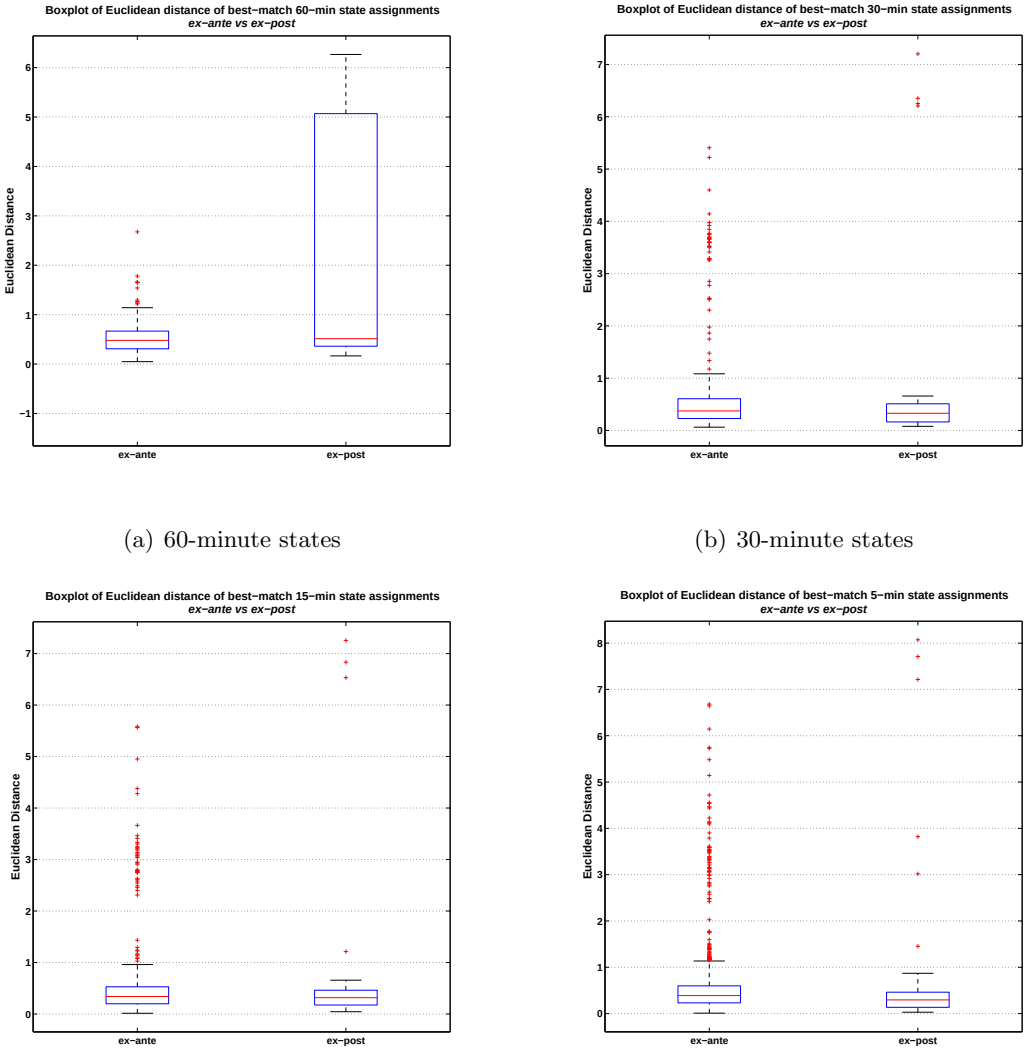


FIGURE 6.16: Estimated 5-minute clusters using identified state signature vectors. The Euclidean distance is calculated between each temporal period's feature vector and the state signature vectors. Cluster index assignment is based on the state signature vector which yields the minimum distance.

	state_{t+1}																								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
1	0.04	0.08	0.17	0.06	0.12	0.14	0.07	0.00	0.19	0.00	0.10	0.01	0.00	0.01	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.08	0.01	0.16	0.14	0.08	0.23	0.03	0.01	0.13	0.01	0.04	0.02	0.01	0.03	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.01
3	0.04	0.06	0.12	0.12	0.11	0.33	0.05	0.02	0.07	0.00	0.02	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.04	0.07	0.24	0.03	0.17	0.21	0.02	0.02	0.08	0.01	0.05	0.00	0.00	0.01	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.00
5	0.04	0.03	0.16	0.13	0.06	0.16	0.08	0.03	0.23	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.07	0.07	0.32	0.12	0.08	0.09	0.02	0.01	0.13	0.02	0.03	0.01	0.00	0.02	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.14	0.07	0.24	0.04	0.23	0.10	0.04	0.01	0.05	0.00	0.02	0.00	0.00	0.01	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.02	0.05	0.33	0.10	0.10	0.18	0.03	0.03	0.10	0.03	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9	0.11	0.04	0.15	0.04	0.20	0.30	0.04	0.02	0.03	0.00	0.03	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	0.06	0.12	0.24	0.00	0.00	0.18	0.06	0.06	0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	0.19	0.01	0.26	0.21	0.07	0.13	0.00	0.00	0.04	0.01	0.03	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
state_t **12**	0.11	0.11	0.00	0.00	0.22	0.22	0.00	0.22	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
13	0.00	0.33	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
14	0.03	0.03	0.17	0.00	0.10	0.37	0.00	0.03	0.10	0.00	0.07	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00
15	0.00	0.00	0.33	0.00	0.33	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
16	0.00	0.00	0.20	0.40	0.20	0.00	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
17	0.20	0.00	0.20	0.20	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
18	0.00	0.12	0.24	0.24	0.04	0.12	0.04	0.08	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.04	0.00	0.00	0.00
19	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	0.14	0.14	0.14	0.14	0.00	0.14	0.00	0.00	0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
21	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
22	0.00	0.00	0.25	0.00	0.00	0.50	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
23	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
24	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TABLE 6.8: Empirical 1-step transition probability matrix for 5-minute states, based on identified temporal cluster configuration. State transitions with a probability > 0 are highlighted in green.



(a) 60-minute states

(b) 30-minute states

(c) 15-minute states

(d) 5-minute states

FIGURE 6.17: Measuring the stability of the online state assignment algorithm out-of-sample. Given that the state assignment of an online FV is based on the minimum Euclidean distance to predetermined SSVs, we compute the *best match* distance for each of the FVs in a sample and use a boxplot to visualise the empirical distribution. In this figure, we compare the *ex-ante* (01-Nov-2012 to 30-Nov-2012, same period used for SSV estimation) and *ex-post* (03-Dec-2012 to 07-Dec-2012, one week after SSV estimation window) periods.

6.10 Identifying high-frequency persistent states using *event-time clustering*

While we only consider *calendar time* states above and in the use-case in Chapter 8, we demonstrate here how the concept can be extended to *event time*. By choosing a particular feature of a stock to represent the *system clock*, we can extract a temporal state representation through the lens of the stock considered. In particular, we consider *traded volume of a particular stock* as the event governing the system clock, e.g. if we seek a state representation appropriate for trading small volumes of stock AGL (Anglo American), we specify a traded volume bin of, say, 1000 shares. Each time 1000 shares of AGL has traded, the clock for the entire system ticks. We thus move from fixed *calendar time* bins to fixed *AGL traded volume* bins. We conjecture that *event-time* states will capture the appropriate information in LOB dynamics which is suitable for the operating scale of a machine trading algorithm. To demonstrate this concept, using *traded volume* to govern our system clock, we construct *100 000 SHF* event-time clusters in Figure 6.18 and *100 000 AGL* event-time clusters in Figure 6.20. We notice that there are more dark green and dark red nodes here, compared to the *calendar time* results in Section 6.9.4. This is consistent with the notion of higher liquidity during morning and afternoon trading sessions, with higher traded volumes causing the clock to tick faster. We also note that AGL is a more liquid stock than SHF, thus for a given traded volume bin size (100 000), there are more periods for AGL than SHF, as indicated by the higher number of nodes in Figure 6.20. Figures 6.21 and 6.19 show the associated SSVs from significant states. Importantly, when constructing *event-time* states, we notice that the resultant cluster sizes still yield significant power-law fits, as shown in Figure 6.22. Here, the p-value for the *100 000 SHF* cluster size fit is 0.16885 and for the *100 000 AGL* fit is 0.96315. This is consistent with our argument in Section 6.7, where sampling the system at *event-time* scale yields statistically significant power-law fits, indicating that the system is at or near criticality. Since we have observed significant power-law fits at all *calendar* and *event* times scales that we have sampled the system, this provides more evidence towards the notion of financial markets as complex adaptive systems.

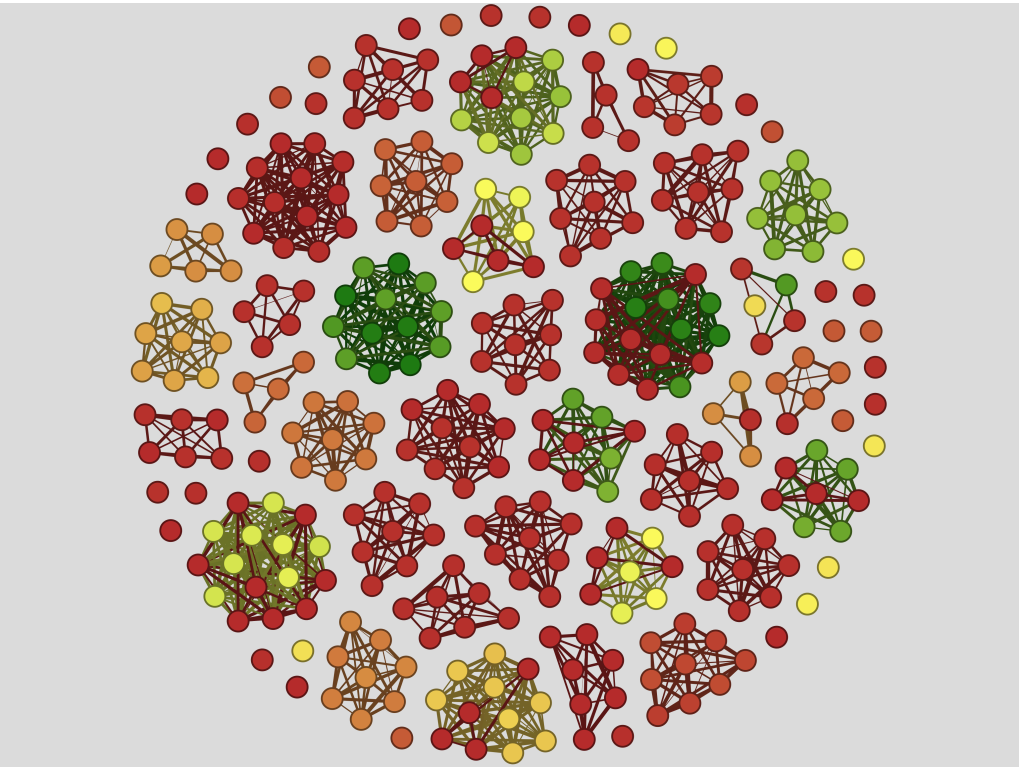


FIGURE 6.18: JSE TOP40 event-time clusters for the period 01-Nov-2012 to 30-Nov-2012. Each node represents a **traded volume of 100 000 SHF shares**, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership.

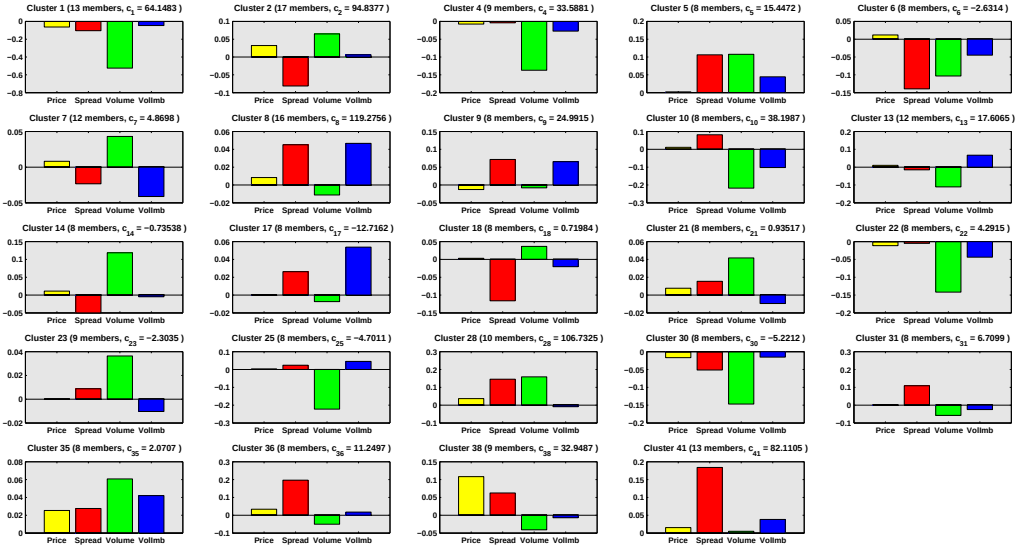


FIGURE 6.19: JSE TOP40 **100 000 SHF volume** cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses.

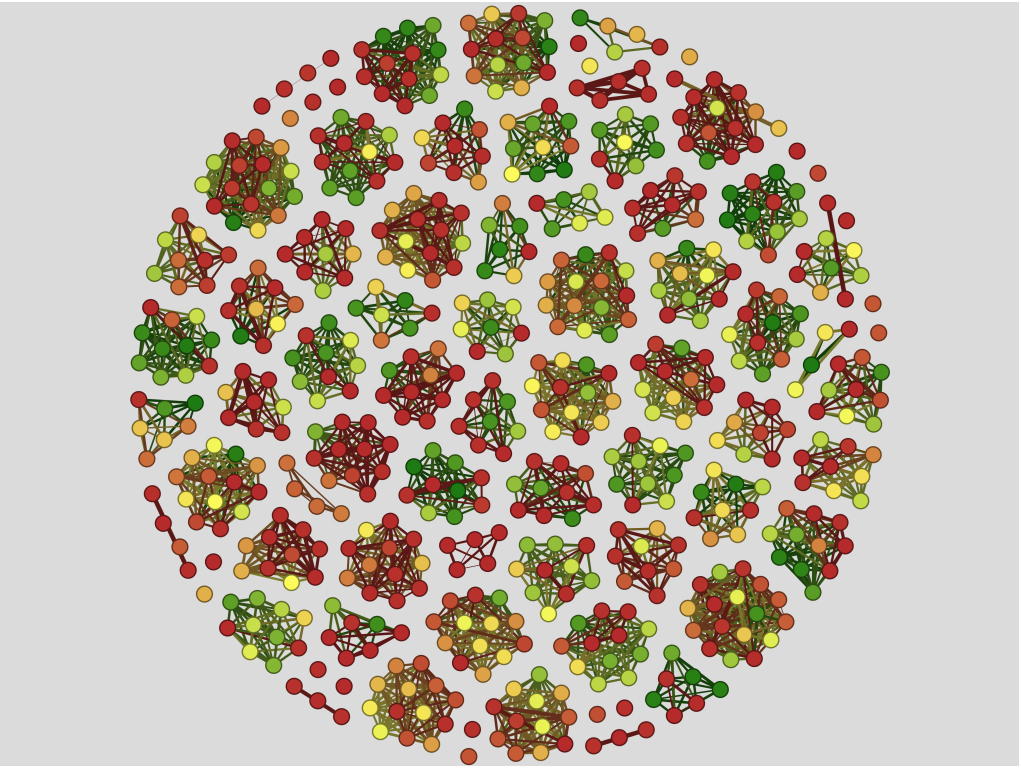


FIGURE 6.20: JSE TOP40 event-time clusters for the period 01-Nov-2012 to 30-Nov-2012. Each node represents a **traded volume of 100 000 AGL shares**, with the colour shading indicating the time-of-day (Morning = green, Lunch = yellow, Afternoon = red) and node connectedness indicating cluster membership.

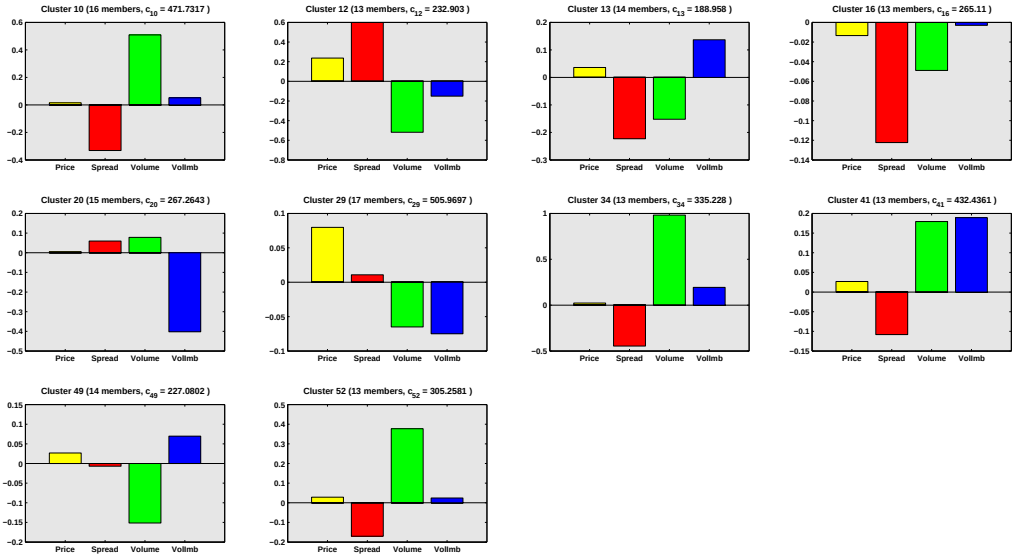
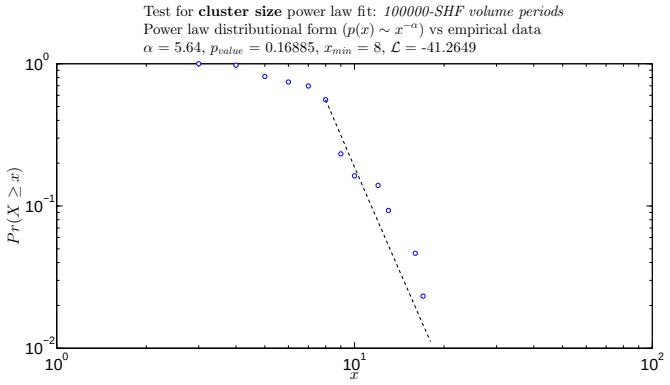
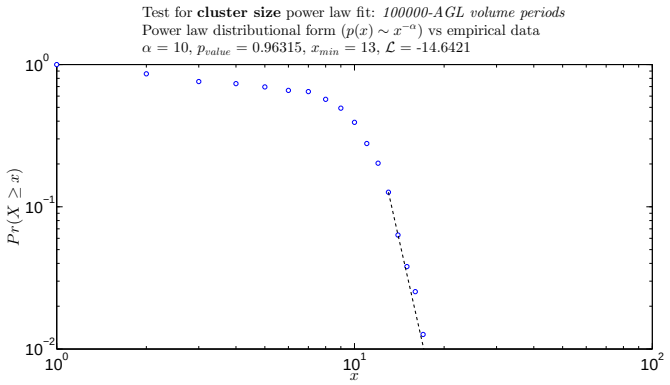


FIGURE 6.21: JSE TOP40 **100 000 AGL volume** cluster state signature vectors for the period 01-Nov-2012 to 30-Nov-2012. Each plot illustrates the average change in trade price, spread, trade volume and quote volume imbalance across member periods and stocks for each of the clusters with a size $\geq x_{min}$ from the truncated power-law fit. Cluster size and intra-cluster correlation are shown in parentheses.



(a) *100 000 SHF volume* cluster sizes



(b) *100 000 AGL volume* cluster sizes

FIGURE 6.22: Testing conjecture of power law fit for varying time scale cluster sizes, applying the Clauset, Shalizi and Newman algorithm [58]. α indicates the scaling parameter of the proposed fit, p_{value} indicates the p-value from a Kolmogorov-Smirnov test for the goodness-of-fit of the proposed model to the data, x_{min} indicates the lower-bound for the power law fit and $\mathcal{L}$ is the log-likelihood of the data ($x \geq x_{min}$) under the power law fit.

6.11 Some remarks

In this chapter, we have outlined a novel approach for the unsupervised detection of intraday temporal market states at varying time scales, as well as a proposed mechanism for significant state selection and online state estimation. Using the maximum likelihood approach of [103], we show that the technique can be used to cluster temporal periods as objects based on market microstructure feature performance. A high-speed PGA was used for cluster detection, with a computation time conducive to overnight or even intraday calibration of market states. A study of temporal cluster configurations and power-law fits to 60-minute, 30-minute, 15-minute and 5-minute time scales revealed scale-specific system behaviour, motivating the need for scale-specific state space reduction for optimal planning of participating trading agents. The proposed scheme for online state detection suggested the use of SSVs to capture the market activity signature of each identified state, with a simple distance metric of the prevailing FV to

determine the state index. We showed that the online state detection scheme can be used to enumerate and update 1-step transition probability matrices, which can be used for optimal planning in the high-frequency trading domain. We considered the stability of the algorithm *ex-post* and found that we could reliably determine 30-minute, 15-minute and 5-minute states using the proposed algorithm, whereas 60-minute states were less stable.

A preliminary study of *event-time* clusters reveals its feasibility as an extension of this work, to construct state representations *through the lens of the stock considered*, providing a temporal state evolution appropriate for the trading scale of the agent.

We also note the recent work of Föllmer [87] where, in the context of risk measures, he uses an analogy to Gibbs measures in statistical mechanics to show that only entropic local risk measures permit a global risk measure which is uniquely determined by the local specification. We have shown that a spin glass model clustering approach, inspired by statistical physics and using correlation as a ferromagnetic interaction term, can be used to determine scale-specific temporal states, and that there is no apparent trivial hierarchical aggregation of state behaviour. This may be linked to the problem of *local consistency* of risk measures considered by Föllmer, where he conjectures that the non-uniqueness of a global measure contributes in some way to perceived systemic risk or non-trivial aggregation. One application of Föllmer's findings may combine the entropic regime restriction with the temporal state detection mechanism, to determine whether there is some hierarchical state behaviour across scales present in the system. This should be explored in future work.

Chapter 7

Using order book resiliency to control agent actions

7.1 Overview

This chapter introduces the multivariate Hawkes process as a candidate governing process for the arrival times of limit order book events. We provide a scheme which enumerates empirical point processes for key liquidity demand and replenishment events, permitting the calibration of an appropriate Hawkes process and quantifying order book resiliency. By considering volume-conditional liquidity-demand events, together with a sum-of-exponentials kernel, we are able to quantify the branching ratios of liquidity replenishment intensities following trades of varying size. The scale-specific state representation proposed in Chapter 6 assumes it is reasonable to treat the temporal evolution of the system as exogenous to the trading agent. The resiliency study allows us to constrain the action set of the trading agent such that the order book is assumed to be resilient with respect to the maximum permissible order size. This ensures that an appropriate action set is determined for the stock concerned, such that the assumption of exogenous evolution of order book dynamics, for state representation purposes, is validated. The contribution is captured in the following two papers:

D. Hendricks, M. Harvey. *Reconciling order book resiliency and price impact.* Working paper, 2016. [123]

R. Martins, D. Hendricks. *The statistical significance of multivariate Hawkes processes fitted to limit order book data.* Working paper (submitted to Journal of Applied Probability, under review), 2016. [174]

7.2 Modelling order book resiliency

7.2.1 Multivariate Hawkes process for limit order book events

Alan Hawkes introduced a class of multivariate point processes with a stochastic intensity vector, incorporating event-occurrence clustering behaviour in a coupled system [120]. Initial applications used calibrated Hawkes processes to measure the conditional intensities of earthquakes and aftershocks, based on recorded data [204, 205, 240]. In financial markets, empirical studies of market microstructure have highlighted apparent clustering of limit order book events at tick scale, with some event intensities exhibiting dependent behaviour [4, 36, 109].

The simplest Hawkes processes are univariate, considering a single event type and its temporal dependence on prior events. These models contain an *exogenous* or *baseline intensity* component, which corresponds to the intensity of events that is not influenced by the occurrence of prior events, and an *endogenous* intensity component, which corresponds to the increased intensity of *child* events that occurs after an exogenous *parent* event, thus capturing the clustering of events that one often expects in financial data. A multivariate Hawkes process, then, extends this to include multiple event types, where we further define *self-excitation* of one event type influencing more of its own type of event, and *mutual-* or *cross-excitation* of one event precipitating the occurrence of other types of events.

Definition 7.1. Multivariate Hawkes process
Consider a point process $N(t)$ such that

$$\mathbb{P}[\triangle N(t) = 1 | N(s)_{s \leq t}] = \lambda(t)\triangle_t + o(\triangle_t) \text{ and}$$
$$\mathbb{P}[\triangle N(t) > 1 | N(s)_{s \leq t}] = o(\triangle_t).$$

For a multivariate point process $N(t) = \{N_r(t) : r = 1, ..., R\}$, the rth intensity function of the R-variate mutually-exciting Hawkes process is given by

$$\lambda_r(t) = \mu_r(t) + \int_{-\infty}^{t} \sum_{i=1}^{R} \phi_{r,i}(t-u) dN_i(u)$$

where

$\mu_r(t)$ is the time-dependent baseline intensity for the rth event type

$\phi_{r,i}(t)$ is the kernel function, which encodes the dependency on prior events of type i and satisfies the following conditions [24]:

1) Component-wise positive, i.e. $\phi_{r,i}(t) \geq 0$ *for each* $1 \leq r, i \leq R$

2) Component-wise causal, i.e. if $t < 0$, *then* $\phi_{r,i}(t) = 0$ *for each* $1 \leq r, i \leq R$

3) Each component belongs to the space of L^1*-integrable functions.*

The point process $N(t)$ and intensity vector $\lambda(t)$ together characterise the Hawkes process.

Bacry et al. provide a comprehensive review article highlighting the many applications of Hawkes processes in finance [24]. Bowsher considered one of the first applications, where a bivariate point process of the timing of a stock's trade price and mid-quote changes was used to model volatility clustering on the New York Stock Exchange [46]. A key phenomenon investigated using Hawkes processes is *endogeneity* in financial markets [83, 84, 112, 113]. Empirical observation reveals that, in certain instances, market prices change too quickly to be strictly attributed to the flow of pertinent information, and thus evade explanation in classic economic theory [46]. By considering the ratio of exogenous parent events to endogenous events, it is possible to obtain a measure of market reflexivity [83].

A number of studies by Degryse et al., Large and Bacry et al. have used a multivariate Hawkes process to quantify the *resiliency* of a limit order book, viz. the propensity for quote replenishment following a liquidity demand event [20, 69, 159]. By characterising and extracting key liquidity demand and replenishment events from a limit order book, and using an appropriate choice of kernel to encode temporal dependence of events, Large claims it is possible to use a calibrated Hawkes process to calculate the probability and expected half-life of quote replenishment following a liquidity demand event [159].

We identify key aggressive liquidity demand and replenishment events, enumerating empirical event point processes for model calibration and quantification of LOB resiliency. We first investigate an appropriate kernel shape for the multivariate Hawkes process which provides a significant fit to the empirical data extracted from the Johannesburg Stock Exchange (JSE). We note that this work is very much in the same spirit as [157], however we are determining the statistical significance of *multivariate* Hawkes processes fitted to LOB event data. We then use this model to quantify LOB resiliency, but use volume-conditional liquidity demand events. The calibrated branching ratios of the

multivariate Hawkes process can be used to assess the resiliency of the order book with respect to market orders of varying volumes. Given that we need to measure the effects of our trading agent's interactions with the system, and the debate around accurately measuring permanent or transient price impact [42], we rather choose to use the resiliency study to inform the actions chosen by the trading agent. By assessing the intensity of quote replenishment events and determining the trade size of liquidity demand events which do not have a commensurate resilient response in quote replenishment, we conjecture it is possible to assign a maximum permissible trade size for our trading agent which is not expected to materially alter LOB dynamics.

7.2.2 Enumerating empirical event point processes using tick data

Typical limit order book (LOB) events include trades, new quotes, quote modifications and quote cancellations [4]. Following the suggestions by Large [159] and Biais et al. [36] and taking into account the nature of our data, we define 4 key *aggressive* liquidity demand and replenishment event types which will be used to characterise order book resiliency:

- **Type 1:** *A buy trade that moves the ask*
  The first of the two liquidity demand events, we define an aggressive buy trade as one where the trade price is greater than the best ask price, or it is equal to the best ask, but the volume is greater than that available at the current best ask. Such trades are considered aggressive since they materially alter the shape of the limit order book, pushing the best ask price higher, widening the spread and removing liquidity. Formally, with P representing the trade price, A the prevailing best ask, and V_P, V_A the respective volumes, we express this event type as the following subset of limit order book events:

$$(P > A) \cup \big[(P = A) \cap (V_P \geq V_A)\big]$$

- **Type 2:** *A sell trade that moves the bid*
  The second of the two liquidity demand events is the aggressive sell trade, where the trade price is lower than the current best bid price, or it is equal to the best bid price, but the volume of the trade is greater than that available at the best bid. Similarly, this trade is considered aggressive since it alters the structure of the limit order book by pushing the best bid down, widening the spread and removing liquidity. Referring to the aforementioned notation, only adding that B refers to the best available bid price, we formally characterise aggressive sell trades as the

following subset:

$$(P < B) \cup \big[(P = B) \cap (V_P \geq V_B)\big]$$

- **Type 3:** *A bid between the quotes*
  The first of the two resiliency events, the aggressive bid is a bid between the current best bid and ask. It is considered aggressive since it alters the structure of the limit order book, pushing up the best bid, reducing the spread and providing liquidity through more competitive prices and added volume. With B_* being the incoming bid quote, and B_p the prevailing best bid, we formally characterise the aggressive bid using similar notation to before, as the following subset:

$$(B_* > B_p)$$

- **Type 4:** *An ask between the quotes*
  The second of the two resiliency events, an aggressive ask is defined as an ask quote between the current best bid and ask, considered aggressive since it narrows the gap between the prevailing best bid and ask, providing liquidity. With A_* being the incoming ask quote, and A_p the prevailing best ask, we formally characterise the aggressive ask quotes as the following subset:

$$(A_* < A_p)$$

While not critical for our study of LOB resiliency, we also identify and classify the following *passive* event types for completeness:

- **Type 5:** *Passive buy trade*
  These are buy trades which do not negatively affect LOB liquidity, as they do not affect the prevailing spread. If P is the trade price, A the prevailing ask quote, and V_P and V_A their respective volumes, passive buy trades will be classified as:

$$\big[(P = A) \cap (V_P < V_A)\big]$$

- **Type 6:** *Passive sell trade*
  These are sell trades which do not negatively affect LOB liquidity, as they do not affect the prevailing spread. If P is the trade price, B the prevailing bid quote, and V_P and V_B their respective volumes, passive sell trades will be classified as:

$$\big[(P = B) \cap (V_P < V_B)\big]$$

- **Type 7:** *Passive bid quote*
  These are bid quotes which enter the LOB at a level higher than level-1, and hence do not affect the prevailing spread. If B_* is the incoming bid quote and B_p is the prevailing bid quote, passive bid quotes will be classified as:

$$(B_* < B_p)$$

- **Type 8:** *Passive sell quote*
  These are ask quotes which enter the LOB at a level higher than level-1, and hence do not affect the prevailing spread. If A_* is the incoming ask quote and A_p is the prevailing ask quote, passive ask quotes will be classified as:

$$(A_* > A_p)$$

Figures 7.1, 7.2 and 7.3 illustrate the measured empirical intensities for the 4 key event types, demonstrating event clustering and mutual-excitation over typical morning, mid-day and afternoon periods.

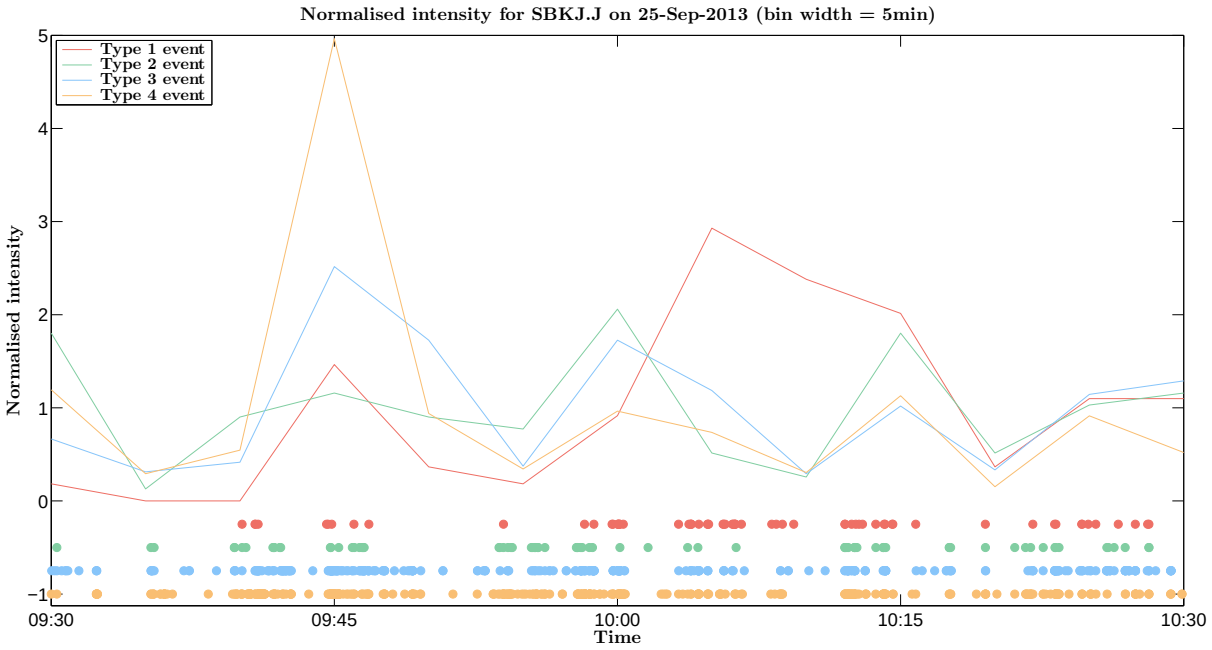


FIGURE 7.1: Empirical event intensities for SBK of each of the 4 key event types over a **morning period**, demonstrating event clustering and mutual-excitation. The dots show the arrival times of the events and the lines show the 5-minute event intensities.

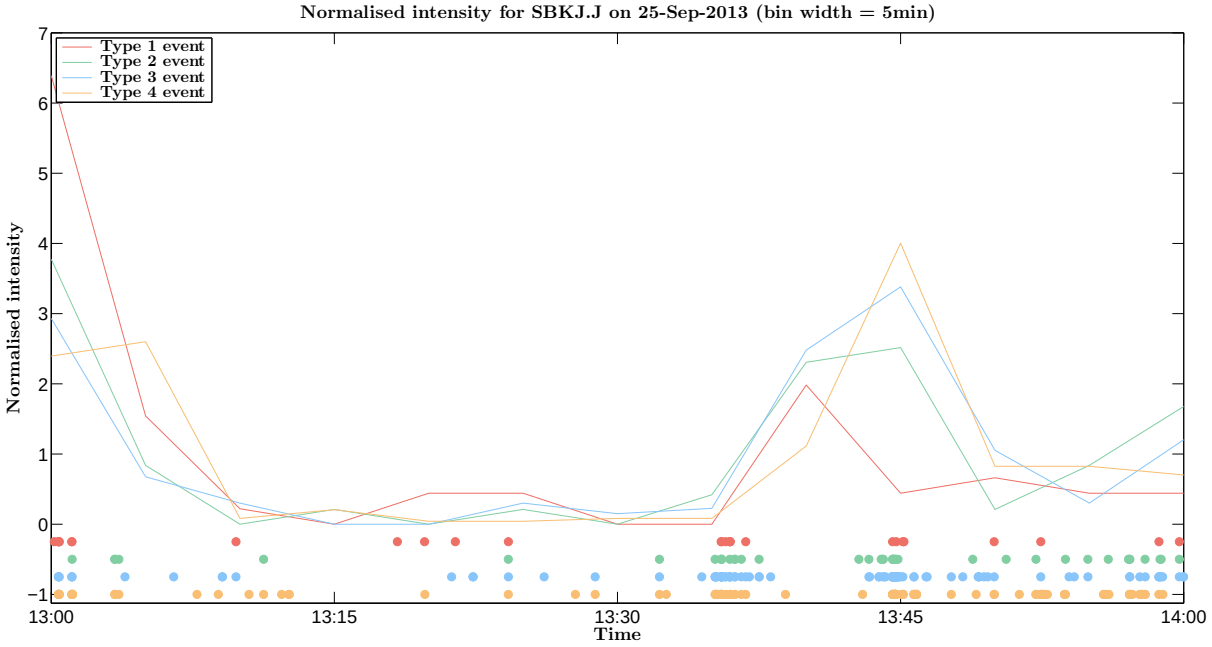


FIGURE 7.2: Empirical event intensities for SBK of each of the 4 key event types over a **midday period**, demonstrating event clustering and mutual-excitation. The dots show the arrival times of the events and the lines show the 5-minute event intensities.

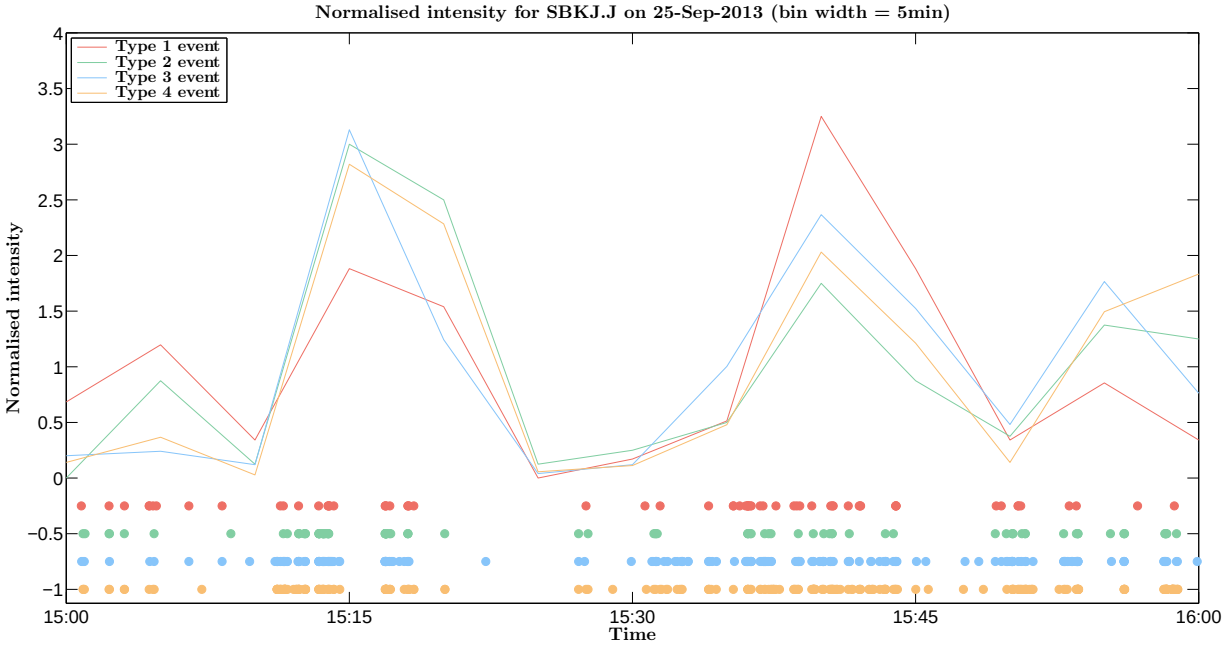


FIGURE 7.3: Empirical event intensities for SBK of each of the 4 key event types over an **afternoon period**, demonstrating event clustering and mutual-excitation. The dots show the arrival times of the events and the lines show the 5-minute event intensities.

7.2.2.1 Volume-conditional liquidity demand point processes

In order to reconcile the LOB resiliency with our goal of controlling the trading agent's submitted market order quantity, we extend the classification of aggressive liquidity

demand events above to account for trade size. The goal is to quantify resilient quote replenishment following a trade of a particular size by examining the effect on calibrated intensities of the multivariate Hawkes process. This will allow us to measure the maximum permissible order size the trading agent can submit, with a reasonable expectation of resilient quote replenishment by the next trading opportunity.

- **Type 1:** *A buy trade of particular size that moves the ask*
  As above, we define an aggressive buy trade as one where the trade price is greater than the best ask price, or it is equal to the best ask, but the volume is greater than that available at the prevailing best ask. Such trades are considered aggressive since they materially alter the shape of the limit order book, pushing the best ask higher, widening the spread and removing liquidity. Formally, with P representing the trade price, A the prevailing best ask, and V_P, V_A the respective volumes, v_i and v_{i+1} the volume bounds for the trade, X as the volume bin increment and $\mathcal{V}$ as the maximum trade volume, we express this event type as the following subset of limit order book events:

  $$\Big[(P > A) \cup \big[(P = A) \cap (V_P \geq V_A)\big]\Big] \cap (v_i \leq V_P < v_{i+1}) \text{ for } v_i = \{0, X, 2X, ..., \mathcal{V} - X\}$$

- **Type 2:** *A sell trade of particular size that moves the bid*
  An aggressive sell trade is one where the trade price is lower than the current best bid price, or it is equal to the best bid price, but the volume of the trade is greater than that available at the best bid. Similarly, this trade is considered aggressive since it alters the structure of the limit order book by pushing the best bid down, widening the spread and removing liquidity again. Referring to the aforementioned notation, only adding that B refers to the best available bid price, we formally characterise aggressive sell trades of a particular size as the following subset:

  $$\Big[(P < B) \cup \big[(P = B) \cap (V_P \geq V_B)\big]\Big] \cap (v_i \leq V_P < v_{i+1}) \text{ for } v_i = \{0, X, 2X, ..., \mathcal{V} - X\}$$

Figures 7.4 and 7.5 show the measured empirical intensities for volume-conditional aggressive liquidity demand events. We have chosen 4 candidate volume bins of equal width to group trade events of similar size.

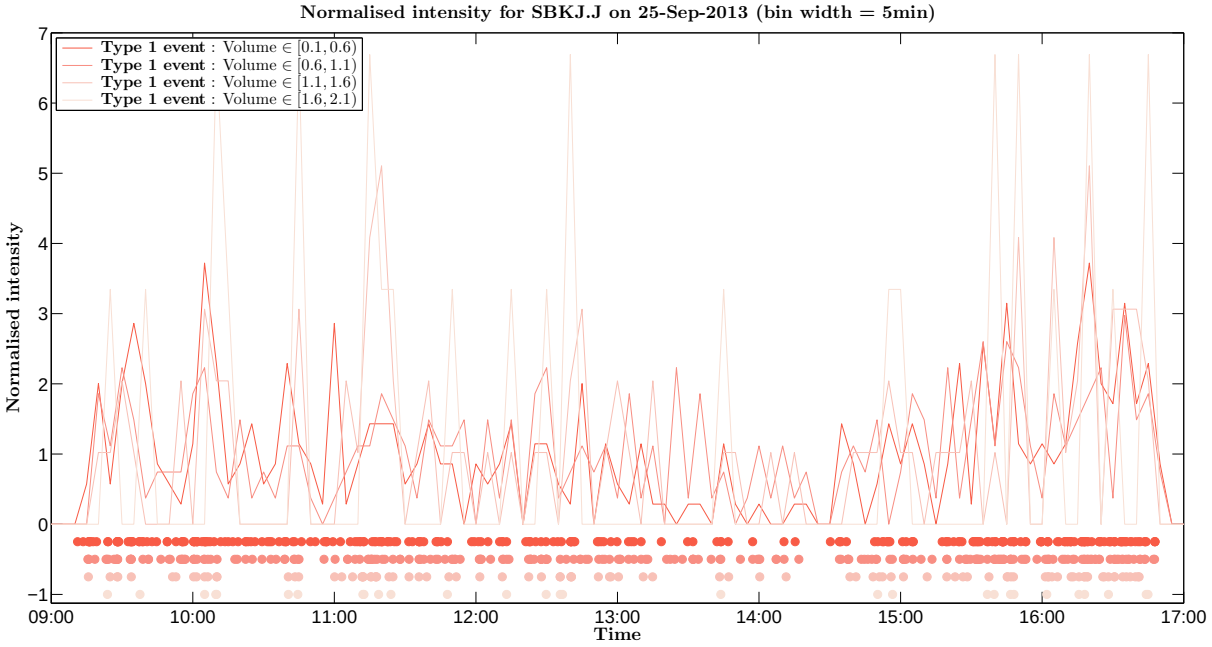


FIGURE 7.4: Empirical **volume-conditional Type-1 event** intensities for SBK for 4 candidate volume bins over a typical trading day. The dots show the arrival times of the events and the lines show the 5-minute event intensities.

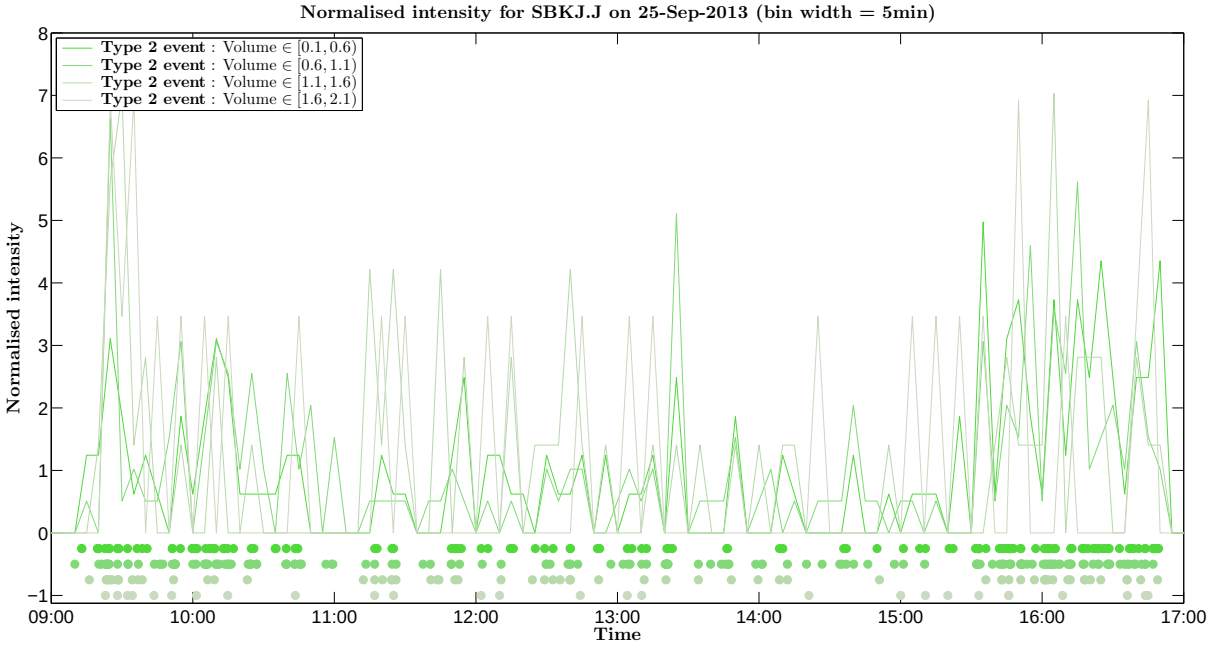


FIGURE 7.5: Empirical **volume-conditional Type-2 event** intensities for SBK for 4 candidate volume bins over a typical trading day. The dots show the arrival times of the events and the lines show the 5-minute event intensities.

7.2.3 Candidate kernels for encoding temporal dependence

A number of kernels have been proposed to model temporal dependence of events, informed by the application or the dynamics of the data being modelled [112]. While,

in principle, any kernel satisfying the conditions in Definition 7.1 can be used, four candidate kernels have typically been considered for financial applications [24, 157, 159]:

- *Sum-of-exponentials*:

$$\phi_M(t) = \sum_{i=1}^{M} \alpha_i e^{-t/\tau_i}$$

  where M is the number of exponentials, α_i is the amplitude of the ith kernel and τ_i is the timescale of the ith kernel. The *branching ratio* is calculated as $n = \sum_{i=1}^{M} \alpha_i \tau_i$.

- *Approximate power law*:

$$\phi_M(t) = \frac{n}{Z} \sum_{i=1}^{M-1} a_i^{-(1+\epsilon)} e^{-\frac{t}{a_i}}$$

  where $a_i = \tau_0 m^i$, M is the range of approximation and m its precision. Z is defined such that $\int_0^\infty \phi_M(t)dt = Z$, n is the *branching ratio*, ϵ is the tail exponent and τ_0 is the smallest timescale.

- *Approximate power law with short lag cut-off* [113]:

$$\phi_M(t) = \frac{n}{Z}\Big(\sum_{i=1}^{M-1} a_i^{-(1+\epsilon)} e^{-\frac{t}{a_i}} - S e^{-\frac{t}{a_{i-1}}} \Big)$$

  where the definition is the same as the *approximate power law*, with the addition of a smooth exponential drop for lags shorted than τ_0. S is defined such that $\phi_M(0) = 0$.

- *Lallouache-Challet power law and exponential* [157]:

$$\phi_M(t) = \frac{n}{Z}\Big(\sum_{i=1}^{M-1} a_i^{-(1+\epsilon)} e^{-\frac{t}{a_i}} + b e^{-\frac{t}{\tau}} \Big)$$

  which is the *approximate power law* with an additional exponential term with free parameters b and τ. This permits greater freedom in the structure of time scales [157].

We will consider the *sum-of-exponentials* kernel, as this will allow us to quantify limit order book (LOB) resiliency by measuring the branching ratio for particular liquidity demand/replenishment event pairs, as well as the expected half-life of replenishment. Large made use of a 2-exponential kernel to quantify LOB resiliency [159], however it is unclear whether two exponentials is appropriate for our events dataset. We will thus consider a preliminary goodness-of-fit study for $M = 1, 2$ and 3 to determine the

appropriate form of the kernel, before we proceed with the volume-conditional resiliency study.

7.2.4 Deriving maximum likelihood estimator with sum-of-exponentials kernel

We will consider the exposition shown in [157] and extend it to the multivariate case. The sum-of-exponentials kernel is defined as

$$\phi_M(t) = \sum_{i=1}^{M} \alpha_i e^{-t/\tau_i}.$$

The M term refers to the number of exponentials to be summed for each event type, α is the individual exponential's unscaled intensity, τ refers to the particular timescale associated to the intensity of one exponential, in contrast to the commonly presented β that refers to the multiplicative inverse of τ, being the decay. Importantly, we note that the *branching ratio*, n, of a particular event type is expressed as

$$n = \sum_{i=1}^{M} \alpha_i \tau_i. \tag{7.1}$$

The branching ratio corresponds to the number of child events a parent event is expected to have. A branching ratio greater than one (*super-critical*) will quickly explode with events, a branching ratio equal to one (*critical*) is a special case where a family will live indefinitely without exploding, as long as $\mu = 0$, and a branching ratio less than one (*sub-critical*) refers to a process where each family of clustered events will eventually die out.

To calculate the half-life of a given intensity effect α_i, we solve

$$1/2 = e^{-\frac{t^i_{1/2}}{\tau_i}} \implies t^i_{1/2} = \tau_i \ln(2), \tag{7.2}$$

with the total half-life given by

$$t_{1/2} = \sum_{i=1}^{M} \tau_i \ln(2). \tag{7.3}$$

The sum-of-exponentials kernel thus yields the following intensity function:

$$\lambda_M(t) = \mu + \int_0^t \sum_{i=1}^{M} \alpha_i e^{-\frac{t-u}{\tau_i}} dN(u).$$

In particular, we are considering the four-variate case, since we are interested in key aggressive liquidity demand and replenishment events to quantify resiliency (see Section 7.2.2 below). With $r = 1, 2, 3, 4$ referring to the event type of interest $\dot{r}$, and assuming a time-dependent baseline intensity, μ, we have

$$\lambda_{\dot{r},M}(t) = \mu_{\dot{r}}(t) + \int_0^t \sum_{r=1}^{4} \sum_{i=1}^{M} \alpha_{r,i} e^{-\frac{t-u}{\tau_{r,i}}} dN_{\dot{r}}(u). \tag{7.4}$$

We refer to the known log-likelihood function for Hawkes processes with exponential or power-law kernels [207]:

$$\ln \mathcal{L}\big(t_1, ..., t_n \big| \theta\big) = -\int_0^T \lambda(t|\theta)dt + \int_0^T \ln \lambda(t|\theta) dN(t)$$

noting that

$$\int_0^t h(s)dN(s) = \sum_{t_i<t} h(t_i).$$

Thus we derive the log-likelihood as

$$\begin{aligned}\ln \mathcal{L}_{\dot{r}}(\theta) = &-\int_0^T \mu_{\dot{r}}(t)dt - \sum_{r=1}^{4}\sum_{i=1}^{M} \alpha_{r,i}\tau_{r,i} \sum_{t_j<T} \left(1 - e^{-\frac{T-t_j}{\tau_{r,i}}}\right) \\ &+ \sum_{t_j<T} \ln \left[\mu_{\dot{r}}(t_j) + \sum_{r=1}^{4}\sum_{i=1}^{M} \alpha_{r,i} \sum_{t_{j'}<t_j} e^{-\frac{t_j - t_{j'}}{\tau_{r,i}}}\right].\end{aligned}$$

Using the following recursive relationships [207], assuming that $\dot{r} = 1$:

$$R_{1,i}(j) = e^{-\frac{t_j - t_{j-1}}{\tau_{1,i}}} \big(1 + R_{1,i}(j-1)\big)$$

and, for mutual event types $r = 2, 3, 4$, letting $\tilde{k} = \sup[k'|t_{k'} < t_j]$:

$$R_{r,i}(j) = e^{-\frac{t_j - t_{j-1}}{\tau_{r,i}}} R_{r,i}(j-1) + \sum_{[k'|t_{j-1} \le t_{k'} < t_j]} e^{-\frac{t_j - t_{k'}}{\tau_{r,i}}}$$

Thus, substituting back into the log-likelihood function, we obtain

$$\begin{aligned}\ln \mathcal{L}_{\dot{r}}(\theta) = &-\int_0^T \mu_{\dot{r}}(t)dt - \sum_{r=1}^{4}\sum_{i=1}^{M} \alpha_{r,i}\tau_{r,i} \sum_{t_j<T} \left(1 - e^{-\frac{T-t_j}{\tau_{r,i}}}\right) \\ &+ \sum_{t_j<T} \ln \left[\mu_{\dot{r}}(t_j) + \sum_{r=1}^{4}\sum_{i=1}^{M} \alpha_{r,i} R_{r,i}(j)\right]\end{aligned} \tag{7.5}$$

The log-likelihood function in Equation 7.5 will be implemented for parameter calibration.

7.2.5 Calibration of model parameters

To quantify LOB resiliency, we require the calibration of the μ, α and τ parameters in Equation 7.5. We used MATLAB to develop a non-linear constrained optimisation routine to find the parameters which maximise the likelihood function specified in Equation 7.5. In particular, we used a sequential quadratic programming algorithm to iteratively adjust candidate parameter values until a best approximation to the maximum likelihood estimator is found, within a given tolerance. To promote finding a global solution and stability in algorithm results, we use a genetic algorithm with Equation 7.5 as the objective function to find feasible parameter values to initialise the optimisation routine. This allows us to narrow the search space, before using the optimisation to refine the calibration.

While the univariate Hawkes process can exploit the recursive relationship for the log-likelihood calculation, reducing the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ [157, 207], this advantage does not translate to the multivariate case. We made use of a number of vectorisation enhancements to improve the computational efficiency of the implemented algorithm, as shown in Figures 7.6 and 7.7. These indicate that our implementation scales well as a function of both number of events and number of exponentials in the chosen kernel, compared to a naïve for-loop implementation. The details of the full implementation can be found in a study by Martins and Hendricks [173, 174].

Once the parameters have been calibrated to the events data, we can obtain values for the branching ratio (Equation 7.1) and half-life (Equation 7.3), and the particular expression of Equation 7.4 that defines the intensity of the Hawkes process, which can in turn be used to calculate the residuals for testing.

Before any inferences were made from calibrated parameter values, we verified the accuracy of our implemented calibration scheme. To do this, we performed extensive simulations of multivariate Hawkes processes with known, induced parameters using the *intensity-based approach* promoted by Dassios and Zhao [66, 163], generating a set of events data. The calibration scheme was then used to recover the induced parameters. The successful recovery of induced parameters from simulated data verified the calibration scheme. Details of this can be found in a study by Mazibuko and Hendricks [176].

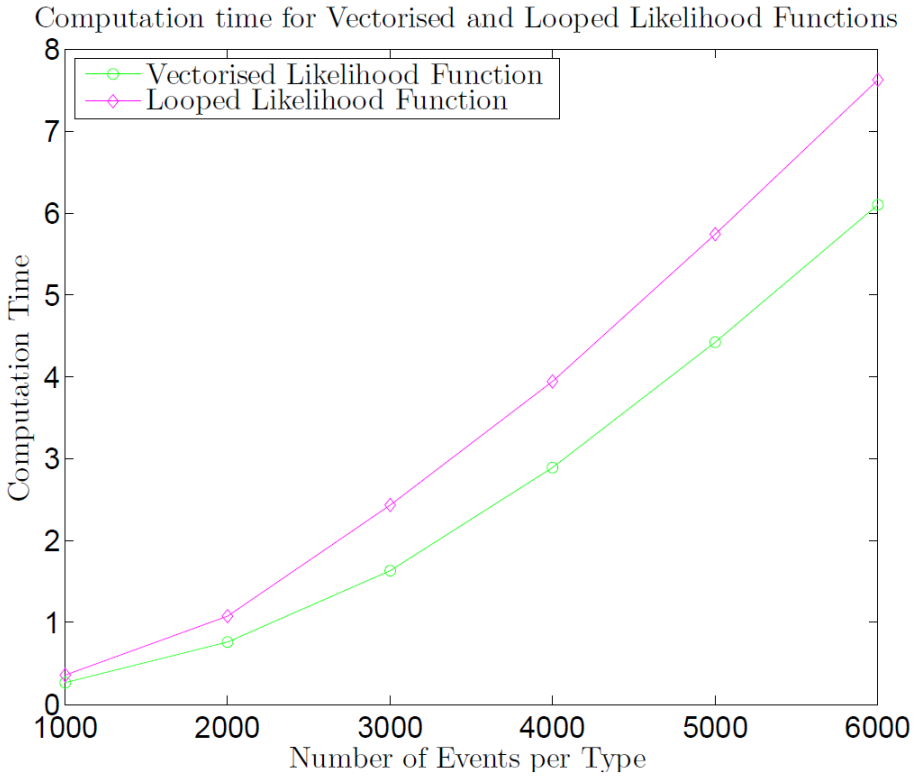


FIGURE 7.6: Computation time (in minutes): vectorisation vs for-loop. Shows average computation time as a function of the number of each event type.

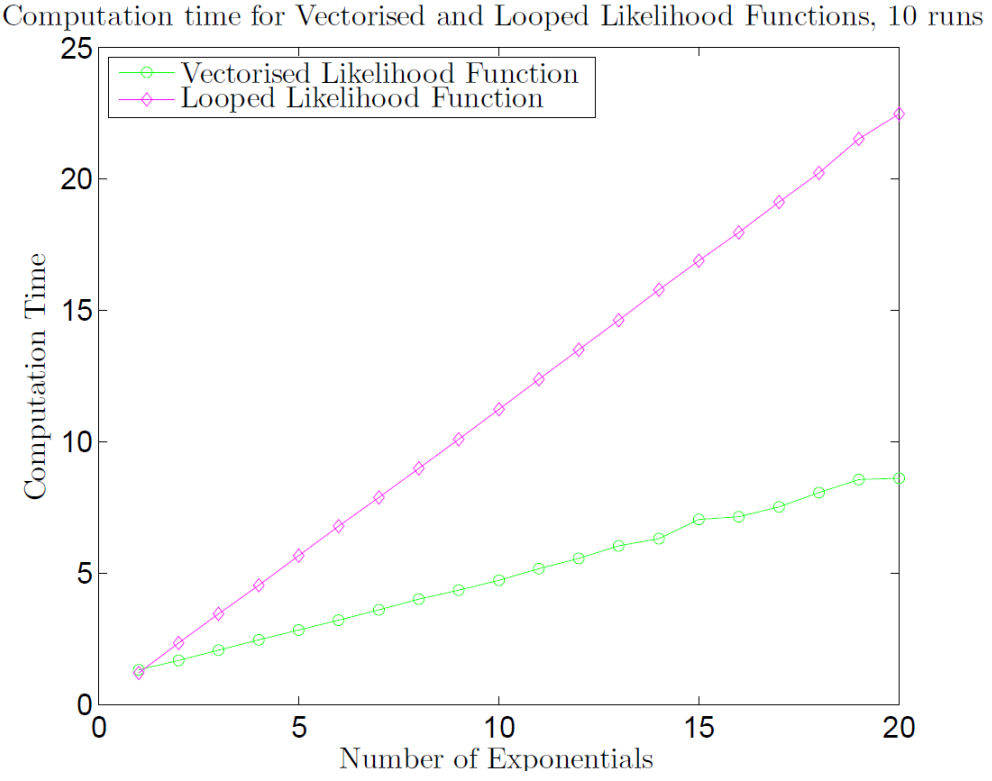


FIGURE 7.7: Computation time (in minutes): vectorisation vs for-loop. Shows average computation time as a function of the number of exponentials (M).

7.2.6 On the choice of M (number of exponentials)

In order to determine the appropriate number of exponentials in our kernel, we performed a number of *goodness-of-fit* tests for $M = 1, 2$ and 3 following calibrations to our dataset of events. We utilise the result in Theorem 7.2 to convert our calibrated Hawkes processes into time-deformed *compensator* functions, viz. sequences of residuals which are exponentially distributed with mean 1.

Theorem 7.2 (Multivariate random time change)**.** *Consider the R sequences of generalised residuals $\{e_i^r(\theta)\}$, $r = 1, ..., R$, where*

$$e_i^r(\theta) := \int_{t_i^{(r)}}^{t_{i+1}^{(r)}} \lambda_r(s, \theta) ds, \tag{7.6}$$

the integrand is the intensity for type r events, and $(t_i^{(r)}, t_{i+1}^{(r)}]$ is interval between adjacent type r events.
When θ is the set of true parameters, each sequence $e_i^r(\theta)$ is an independently distributed exponential random variable with mean 1.

Proof. See Bowsher [46] for a detailed discussion and proof. □

We can then use statistical tests to determine whether the residuals are in fact independent and exponentially distributed with mean 1, where the kernel offering the *best fit* will be used in the analyses that follow. We used four candidate tests to assess the distribution and stationarity of the residuals [113, 157, 159]:

1. *Kolmogorov-Smirnov (KS) test* [152, 225]: This test compares the empirical distribution of residuals to cumulative distribution of an exponential with mean 1. A *non-rejection* of the null hypothesis H_0 indicates empirical residuals are exponentially distributed at the specified significance level.

2. *Excess Dispersion (ED) test* [79]: This test confirms whether the residuals have unit variance, as would be expected if they followed an exponential with mean 1. A *non-rejection* of the null hypothesis H_0 indicates the variance of empirical residuals is 1 at the specified significance level.

3. *Ljung-Box Q (LBQ) test* [165]: This test for stationarity confirms independence of increments, where again, *non-rejection* of the null hypothesis H_0 indicates the residuals are stationary.

4. *Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test* [155]: This is a secondary test for trend stationarity for confirmation of the LBQ results, where *non-rejection* of the null hypothesis H_0 indicates the residuals are trend stationary.

The KS, ED and LBQ statistical tests were performed at a 1% significance level, while the KPSS test for trend stationarity was performed at the 5% significance level, consistent with suggestions by Kwiatowski et al. [155].

Tables 7.1 and 7.2 show the *goodness-of-fit* results for empirical event processes of a candidate stock (GRT) over 2 different periods, for $M = 1, 2$ and 3. We have used to definitions in Section 7.2.2 to extract key aggressive liquidity demand and replenishment processes from the LOB data. Each day is considered as an independent realisation of the multivariate point process, thus a separate calibration is performed for the set of events associated with each day in the dataset. This allows us to construct a distribution of null hypothesis *non-rejections* and p-values for each test, and for each M. For example, column "KS H_0" indicates the proportion of KS tests which confirmed residuals were exponentially distributed with unit mean, whereas column "KS p-value" shows the average p-value for these tests. We see from these results that the 3-exponential kernel offers a better fit to empirical data than the 1- and 2-exponential kernel. This is confirmed by the highest number of null hypothesis acceptances and highest p-values across all statistical tests. We will thus use the 3-exponential kernel in our study of LOB resiliency.

M	**KS** H_0	**KS** p	**ED** H_0	**ED** p	**LBQ** H_0	**LBQ** p	**KPSS** H_0	**KPSS** p
1	0.3421	0.0332	0.2500	0.0628	0.7000	0.2484	0.5658	0.0619
2	0.5658	0.0978	0.4342	0.1311	0.7184	0.2902	0.5395	0.0589
3	0.6184	0.1384	0.5789	0.1720	0.7326	0.2945	0.5526	0.0604

TABLE 7.1: Daily goodness-of-fit test statistics by kernel, **GRT 01 September 2013 to 27 September 2013**

M	**KS** H_0	**KS** p	**ED** H_0	**ED** p	**LBQ** H_0	**LBQ** p	**KPSS** H_0	**KPSS** p
1	0.5000	0.0884	0.3958	0.0928	0.7475	0.2733	0.5729	0.0627
2	0.6771	0.1712	0.5938	0.1784	0.7783	0.2790	0.6146	0.0643
3	0.7917	0.2599	0.7292	0.2658	0.8133	0.2996	0.6250	0.0669

TABLE 7.2: Daily goodness-of-fit test statistics by kernel, **GRT 30 September 2013 to 31 October 2013**

7.2.7 Motivating use of time-dependent baseline intensity

To motivate the use of a time-dependent baseline intensity in our specification in Equation 7.4, we examined the average hourly intensity of our measured empirical event processes. Figure 7.8 shows the average intensity of all events for a candidate stock (GRT)

for each hour of the trading day, averaged over all days in the respective datasets. Both datasets indicate a distinct U-shape for average intensities, which a constant baseline intensity will fail to capture.

Figure 7.9 illustrates the effect of different kernels when assuming a 3-period piecewise linear baseline intensity. We see that a simple *morning*, *noon* and *afternoon* distinction yields calibrated intensities which match the expected U-shape exhibited in the empirical data in Figure 7.8. The shapes of these curves are similar across all three kernels, although we note that as the number of exponentials is increased, we see a decline in the both the mean and variation of the exogeneity. This suggests that the higher number of exponentials permits more explanatory power for cross-exciting effects of events, rather than absorbing these effects into the baseline intensity. Combined with the *goodness-of-fit* tests in Section 7.2.6, this confirms that the drivers of observed event intensities in our data appear to have a higher attribution to self- and cross-exciting effects, as captured by the 3-exponential kernel, than that suggested by fewer exponentials.

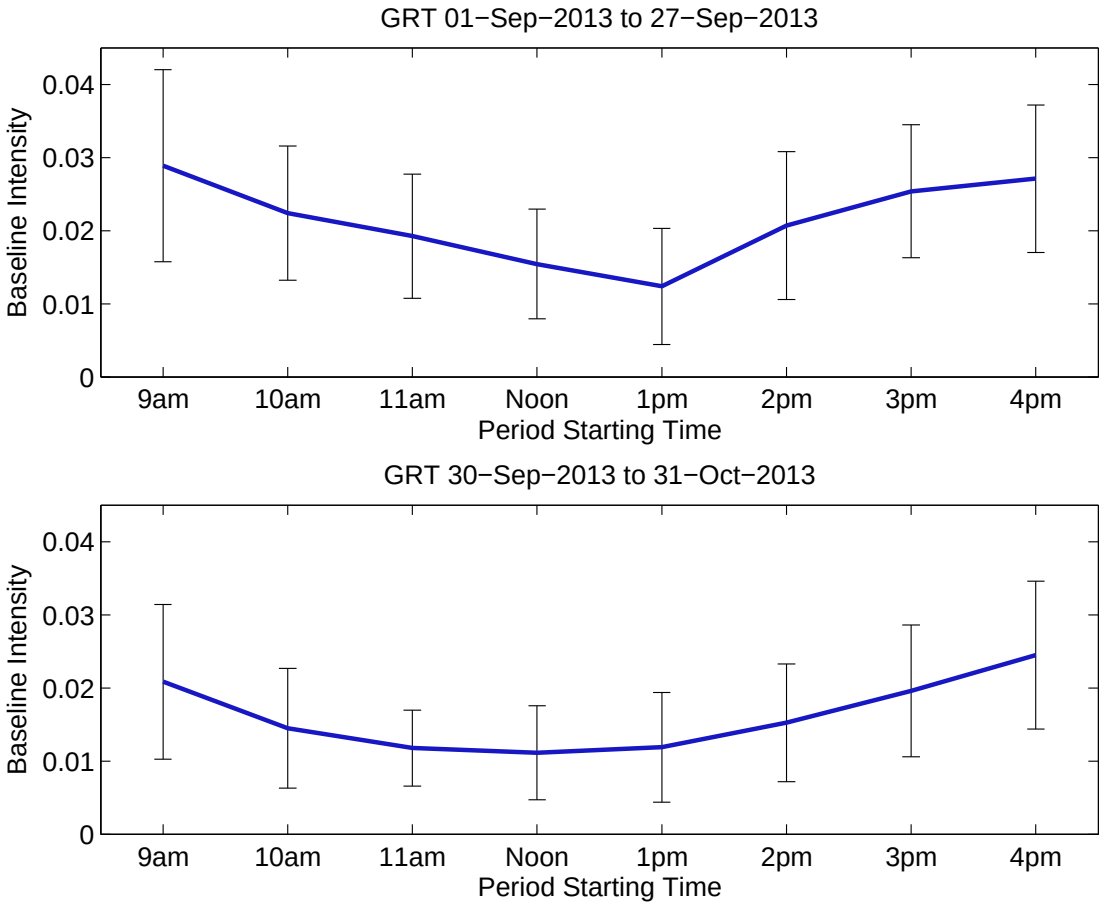


FIGURE 7.8: Average hourly baseline intensity for all events. Blue line indicates average for a given hour, with the error bars reflecting the variation over the days in the sample.

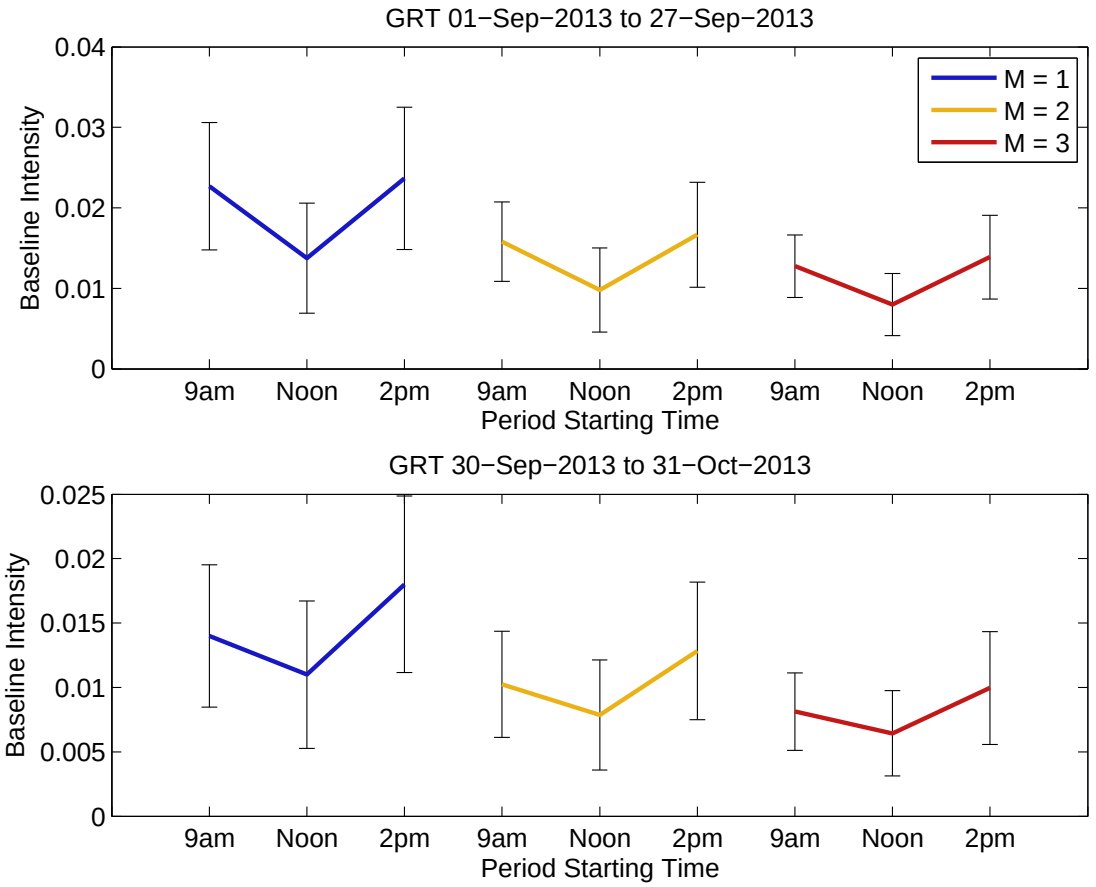


FIGURE 7.9: Baseline intensity by kernel. The trading day is divided into 3 periods (*morning*, *noon*, *afternoon*), with piecewise linear intensity. Coloured lines correspond to mean exogenous intensities for the given periods, calibrated for each kernel, with the error bars reflecting daily variation.

7.2.8 An efficient non-parametric calibration scheme

Our aim in Sections 7.2.6 and 7.2.7 was to find a candidate kernel which offers the best fit to the extracted point processes, using the usual maximum likelihood procedure for calibration and assessing residuals. While it is clear that the sum-of-three-exponential kernel with time-dependent baseline intensity offers the best fit to the empirical data, this requires a 27-parameter MLE calibration. Since each day is treated as an independent realisation of the multivariate point process, the computation time taken for the calibrations required for long-term studies is thus prohibitive and would prevent us from making meaningful inferences for our intended application: finding a trade volume ceiling for our trading agent which the LOB is expected to be able to absorb.

Recent work by Bacry et al. and Kirchner considers efficient non-parametric calibration schemes for the branching ratios of multivariate Hawkes processes [19, 21–23, 150, 215]. Using the fact that a stationary multivariate Hawkes process is completely specified by its first and second order properties, Bacry and Muzy prove that the kernel matrix

is the only causal solution for a Wiener-Hopf system, which is simple to solve with a quadrature scheme [21]. We implemented the scheme described by Bacry and Muzy [21], which uses empirical point processes to recover baseline intensities and kernel shapes, from which the branching ratio can be computed. We used a linear-log binning scheme to estimate the conditional law, with parameters $h_{min} = 1$ and $h_{max} = 2000$, with 30 linear bins and 500 logarithmic bins. Full details regarding the implementation can be found in a study by Hendricks and Harvey [123].

Further work should investigate the stability of calibrated kernels [142], and the efficacy of the non-parametric calibration procedure in its ability to capture the results of the *full* MLE procedure. For the purposes of this thesis, and for computational tractability, we have assumed that the calibrated branching ratios from the non-parametric scheme provide a good approximation to those obtained from the MLE procedure.

7.3 Effect of volume-conditional trade events on quote replenishment intensity

Our intention here is to quantify the impact of trade events of varying size on LOB dynamics, to motivate a scheme for controlling our trading agent's actions and their impact on the system. We show a proof-of-concept here as to how a calibrated resiliency model can be reconciled with permissible trading actions. Using the volume-conditional liquidity demand processes extracted from the LOB data, as described in Section 7.2.2.1, we can quantify the effect of a particular size trade on an associated quote replenishment intensity by assessing calibrated branching ratios of the Hawkes process. Recall that our trading agent, as described in Chapter 5, is executing *market orders*, viz. a buy (sell) trade will absorb available ask (bid) quotes from the LOB, with large trades materially affecting the prevailing spread. If the LOB is not replenished with sufficient ask quotes which improve the spread, this could have adverse cost consequences for the remainder of the trading program. We thus want to ensure that the LOB is expected to be resilient with respect to the maximum permissible trade size that can be executed by the trading agent over the liquidation horizon.

Figures 7.10 and 7.11 show a box-and-whisker plot of distribution of calibrated branching ratios for particular event pairs of volume-conditional Hawkes process. Figure 7.10 quantifies the effect of aggressive buy trade (Type 1) events of varying size on aggressive ask quote (Type 4) replenishment intensity, and Figure 7.11 quantifies the effect of aggressive sell trade (Type 2) events of varying size on aggressive bid quote (Type 3) replenishment intensity. The trade event point processes have been split into six bins on

the basis of normalised trade size, where trade sizes are normalised by the mean trade size. The x-axis labels in the figures indicate the upper bounds of trade volume bins. Thus *bin 1* considers the effect of all trades with a size up to the mean trade size for the entire period considered, *bin 2* considers the effect of trades with a size between the mean trade size and twice the mean trade size, and so on.

We would expect a resilient LOB to exhibit *increasing* branching ratios for quote replenishment intensities following liquidity demand events, i.e. a larger absorption of prevailing quotes should be followed by a commensurate increase in quote replenishment events as a resilient response. We note here that, based on our study of a particular stock (AGL) over the period 01 November 2012 to 30 November 2012, the branching ratios appear to *decrease* as a function of trade size, remaining steady at around 0.1 for trades larger than twice the mean trade size. This suggests that for very large trades (larger than twice the mean), the quote replenishment response remains the same, thus they would have an increasingly adverse effect on LOB dynamics.

It thus seems prudent to *restrict the maximum permissible trade size of our trading agent to twice the mean trade size*, where the mean is measured over a suitable period prior to trading. This would ensure that we can expect a resilient response from the LOB when absorbing quotes via executed market orders. Recall that the scale-specific state representation proposed in Chapter 6 assumes it is reasonable to treat the temporal evolution of the system as exogenous to the trading agent. By restricting the maximum permissible trade size in this way, we can validate this assumption and provides permission for this choice of state representation. This motivates using the SSVs as a public attribute in the agent's state space, maintaining the framework of Chapter 5, with an upper bound on action size of twice the mean trade volume. This will be explored in Chapter 8.

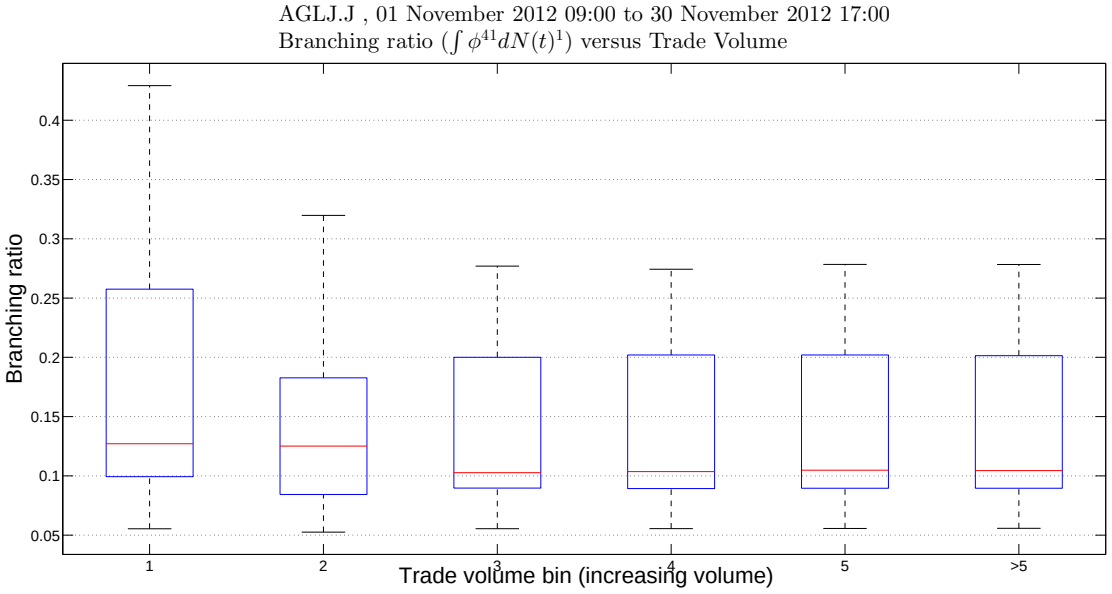


FIGURE 7.10: Boxplot of distribution of calibrated branching ratios of volume-conditional Hawkes process, quantifying the **effect of aggressive buy trade** (Type 1) events of varying size on **aggressive ask quote** (Type 4) replenishment intensity. X-axis labels indicate upper bounds of trade volume bins, where volumes have been normalised by their mean. We thus see the resulting branching ratios for aggressive ask quote intensity, given buy trades with size as increasing multiples of the mean trade size.

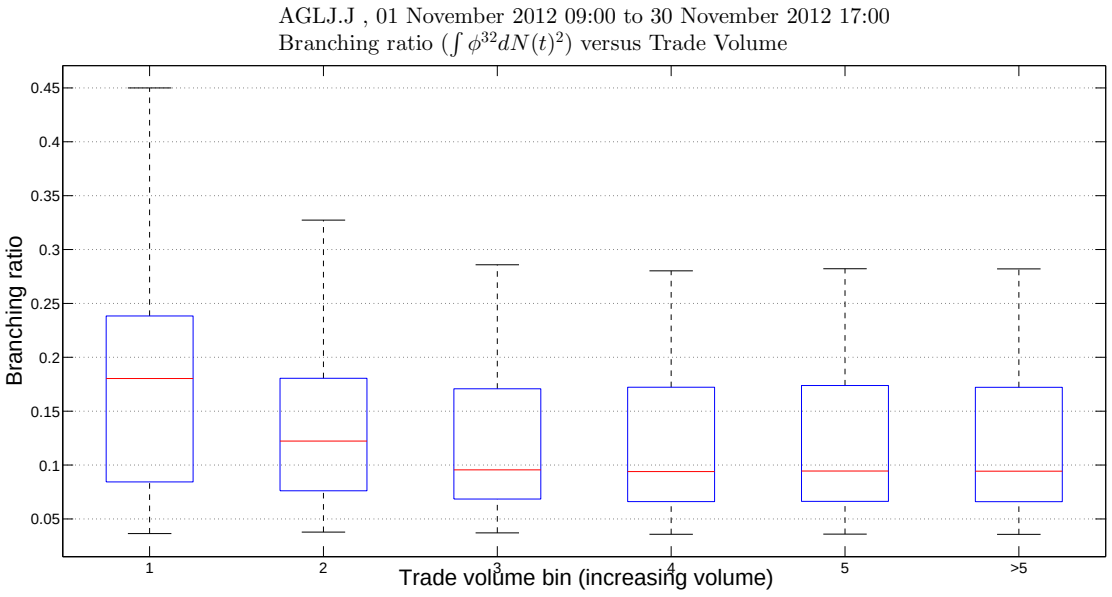


FIGURE 7.11: Boxplot of distribution of calibrated branching ratios of volume-conditional Hawkes process, quantifying the **effect of aggressive sell trade** (Type 2) events of varying size on **aggressive bid quote** (Type 3) replenishment intensity. X-axis labels indicate upper bounds of trade volume bins, where volumes have been normalised by their mean. We thus see the resulting branching ratios for aggressive bid quote intensity, given sell trades with size as increasing multiples of the mean trade size.

7.4 Some remarks

This chapter investigated an appropriate form of the multivariate Hawkes process for modelling LOB resiliency, calibrated to event point processes extracted from the LOB using bespoke classification rules. While results shown here are not extensive, they motivate the use of a 4-variate Hawkes process with time-dependent baseline intensity and sum-of-three-exponential kernel. To investigate the potential effect of our trading agent's actions on system evolution, we used volume-conditional liquidity demand point processes to quantify the effect of *aggressive trade events of varying size* on the intensity of *aggressive quote events*. The branching ratios of calibrated Hawkes models allow us to isolate these effects. We found that, for a particular stock (AGL) and a month's worth of calibrations, the branching ratios appear to *decrease* as a function of trade size, remaining steady at around 0.1 for trades larger than twice the mean trade size. This suggests that for very large trades (larger than twice the mean), the quote replenishment response remains the same, thus they would have an increasingly adverse effect on LOB dynamics.

It thus seems prudent to restrict the maximum permissible trade size of our trading agent to twice the mean trade size, where the mean is measured over a suitable period prior to trading. This would ensure that we can expect a resilient response from the LOB when absorbing quotes via executed market orders.

As a side note, in a recent price impact study by Harvey et al. [115], they found that for many stocks listed on the Johannesburg Stock Exchange, there was an increase in small-volume price impact following a significant fee restructuring at the exchange, which took place in 2013. The fee restructuring removed a price floor for exchange-related trade costs, making fees purely value-based. One could argue that the new scheme enables the profitability of strategies which splitting large orders into many small orders, which could increase the observed intensity of small trade events. We conjecture that the observed increase in small-volume price impact could be explained by a lack of commensurate increase in quote replenishment intensity. This could be assessed using the framework outlined in this chapter, by considering trade intensities and branching ratios of quote-trade pair intensities before and after the fee model change, using a Hawkes model calibrated to point processes consistent with the price impact binning scheme. This will be explored in future work [123]. In addition, it would be prudent to investigate the stability of the calibrated kernels, to determine whether calibrated Hawkes processes in the South African market validate the findings of Jaisson and Rosenbaum [142], where they conclude that only *nearly unstable* Hawkes processes are able to fit the data properly.

Chapter 8

Using detected states and resilient actions to enhance the trade execution algorithm

8.1 Overview

This chapter follows on from Chapter 5, where reinforcement learning is used to determine the optimal sequence of market orders to meet an arrival price trade execution objective. The previous implementation made two key assumptions: 1) *spread* and *volume* sufficiently capture the exogenous temporal evolution of the LOB at calendar scales; 2) agent actions do not affect the exogenous evolution of the LOB. In this chapter, we use the novel state-space reduction technique introduced in Chapter 6 to identify *ex-ante* intraday financial market states, from which low-dimensional *state signature vectors* are extracted to enable online state detection in the learning algorithm. This provides a public state attribute for the learning agent which captures the temporal evolution of the system appropriate for the scale at which the agent interacts with the system. The resiliency study in Chapter 7 provides some validation for assumption (2) above, up to a certain executed trade size. We incorporate this restriction in the permissible action space of the trading agent, and preserve the assumption that LOB dynamics are unaffected by agent actions. In Chapter 9 which follows, we consider a scheme which relaxes this assumption.

The contribution is captured in the following paper:

D. Hendricks. *An online learning algorithm with scale-specific state space enumeration for optimal trade execution in high-frequency markets.* Working paper, 2016. [121]

8.2 Recall the basic reinforcement learning model

In Chapter 5, we introduced a simple discrete-state, discrete-action *Q-learning* algorithm for optimal trade execution with an arrival price benchmark, where the agent learns how to adapt a static liquidation trajectory with respect to prevailing LOB features to improve the program's implementation shortfall.

The *state space* was defined as:

Private Attributes	**Public Attributes**
Elapsed time ($t_n = 1, 2, ..., T$)	Spread ($s_n = 1, 2, ..., B$)
Remaining inventory ($i_n = 1, 2, ..., I$)	Quote volume ($v_n = 1, 2, ..., W$)

TABLE 8.1: State attributes for simple RL trading agent.

The *private attributes* are specific to the trading agent's program, and the *public attributes* are treated as exogenous, unaffected by the trading agent's executed actions. The resolution of the *remaining inventory, spread* and *quote volume* attributes is specified *a priori* by parameters I, B and W.

For the *action set*, we first compute a static liquidation trajectory $(AC_1, AC_2, ..., AC_T)$ using the Almgren-Chriss (AC) [13] approach for a given volume-to-trade (V), fixed time horizon (T) and discrete trading periods $(t = 1, 2, ..., T)$. AC_t represents the proportion of V to trade in period t, such that $\sum_{t=1}^{T} AC_t = V$. We assume that each child order is executed as a *market order* based on the prevailing LOB structure. We would like our learning agent to modify the AC volume trajectory based on prevailing volume and spread characteristics in the market. As such, the possible actions for our agent include:

- β_j = Proportion of AC_t to trade,
- β_{LB} = Lower bound of volume proportion to trade,
- β_{UB} = Upper bound of volume proportion to trade,
- **Action**: $a_{jt} = \beta_j AC_t$, where $\beta_{LB} \leq \beta_j \leq \beta_{UB}$ and $\beta_j = \beta_{j-1} + \beta_{incr}$.

To ensure that the total volume-to-trade is executed over the given time horizon, we execute any residual volume at the end of the trading period with a *market order*.

For the *reward*, we use the difference between the trade price (associated with the market order executing the *action* volume) and the program's *arrival price*, i.e. implementation shortfall. The size of the market order determines how deep to traverse the LOB, with

larger trades typically incurring higher temporary price impact, affecting the trade price. The agent will learn the consequences of each action over the trading horizon, with the ultimate goal of minimising the total trade's *implementation shortfall* by trading more (less) when conditions are favourable (unfavourable).

Given the specification of *states*, *actions* and *rewards*, we can then design a *Q-learning* [243] algorithm for finding an optimal state-action policy in a Markovian domain with unknown dynamics through iterative interactions. We recall that in the tth episode, the agent:

- observes its current state $S_t \in \mathcal{S}$,
- selects and performs an action $A_t \in \mathcal{A}$,
- observes the subsequent state S_{t+1} as a result of performing action A_t,
- receives an immediate reward r_t and
- uses a learning factor α_t, which decreases gradually over time.

Q is updated as follows:

$$Q_{t+1}(S_t, A_t) \quad = \quad Q_t(S_t, A_t) + \alpha_t[r_t + \gamma \max_b Q_t(S_{t+1}, b) - Q_t(S_t, A_t)], \qquad (8.1)$$

where γ is the *discount rate* controlling the importance of future rewards. More details can be found in Chapters 3 and 5. In particular, we considered a *batch RL* agent, where SARSA tuples from a historical dataset were used to *seed* the *Q-matrix* with reasonable values before deploying it online. This was permissible under the assumption of exogenous evolution of public state attributes. We preserve this assumption in this chapter.

8.3 Using temporal state as market attribute

We now consider replacing the *public attributes* in Table 8.1 with the SSVs computed using the methodology in Chapter 6. This allows us to capture the *trade price*, *spread*, *trade volume* and *quote volume imbalance* features, *without the need to specify the resolution for discretisation*. The resolution is implied by the number of critical temporal states identified by the power-law fit to cluster size, as discussed in Section 6.7. The SSVs offer a low-dimensional representation which captures the temporal evolution of the public attributes at the scale at which the agent interacts with the system. Here, we consider calendar-time interactions (5 minute periods), however, as demonstrated in

Section 6.10, we can use the framework to extract a set of SSVs for event-time interactions through the lens of the stock being traded. This will be considered in future work.

8.4 Bounding actions using resiliency

In Chapter 7, we investigated the resiliency of the LOB by quantifying aggressive quote replenishment following trade events of varying size. We conjecture that we can ensure a resilient response of the LOB to trade actions by restricting the maximum permissible market order size of our trading agent to twice the mean trade size, where the mean is measured over a suitable period prior to trading, such as the week preceding the start of the trading program. It is thus straightforward to modify our agent's action space as follows:

- β_j = Proportion of AC_t to trade,
- β_{LB} = Lower bound of volume proportion to trade,
- β_{UB} = Upper bound of volume proportion to trade,
- **Action**: $a_{jt} = \min\{\beta_j AC_t, 2\bar{\mu}_{st_0}\}$, where $\beta_{LB} \leq \beta_j \leq \beta_{UB}$, $\beta_j = \beta_{j-1} + \beta_{incr}$ and $\bar{\mu}_{st_0}$ is the mean buy/sell trade size measured over time $s < t_0$ to the start of the trading program, t_0.

The mean trade size, $\bar{\mu}_{st_0}$, will be measured using only buy trades or only sell trades, based on the intended direction of the trading program. We also assume that this trade size ceiling remains fixed for the trading program.

8.5 On the learning rate

In Section 3.8, we discuss the importance of the learning rate for converging on useful policies fast enough when interacting with a complex adaptive system. The simulated results of Galla and Farmer [95] suggest that, given a payoff correlation amongst competing agents, different learning rates directly affect the nature of the attractor (*Q-matrix*). While it is difficult to estimate the payoff correlation amongst competing trading agents in financial markets, the results of Galla and Farmer highlight the importance of choosing the correct learning rate to ensure learning takes place in a multiple-fixed point regime, or else attempts at learning may be random. We address this issue in a brute-force manner, trying a range of candidate learning rates to verify which one results in

the best *ex-post* performance. A more elegant approach would extend the results of Galla and Farmer to multiplayer games and attempt to identify the qualitative nature of asymptotic learning for trading agents in financial markets. This could be explored in future work.

8.6 Algorithm

Algorithm 3 illustrates the modified procedure from Chapter 5, using computed SSVs as the public state attribute in the learning algorithm. We first specify the trading time-scale (either regular *calendar-time* periods or *event-time* periods), as well as features to extract for each stock. We then compute time-period correlations, identify the temporal cluster configuration, use the power-law criteria to isolate significant states and extract time-scale-specific SSVs for use in the learning algorithm. Given the batch RL setting we consider, we append the *training set* tuples with the associated SSVs as the public state attribute, and proceed with training the *Q-matrix* as before.

Algorithm 3 Q-learner with SSV public state attribute

procedure DETERMINE SCALE-SPECIFIC SSVS(time-scale, features)
    Extract feature returns at the chosen time-scale for all stocks in market
    Calculate time-period correlations
    Calculate temporal cluster configuration
    Fit power-law to cluster size
    Extract time-scale specific SSVs from clusters with size $\geq x_{min}$
end procedure
procedure OPTIMAL STRATEGY(V, T, I, A)
    for Episode 1 to N **do**
        Record reference price at $t = 0$
        for t = T to 1 **do**
            Determine episode's *STATE* attribute (SSV)
            for a = 1 to A **do**
                Determine the action volume a and resulting remaining inventory i
                Set x = (t, i, SSV)
                Calculate IS from trade, $R(x, a)$
                Simulate transition x to y
                Lookup $\max_p Q(y, p)$
                Update $Q(x, a) = Q(x, a) + \alpha U$
            end for
        end for
    end for
    Select the *lowest-IS* action $max_p Q(y, p)$ for optimal policy
end procedure

8.7 Data and Results

8.7.1 Data

We used one month of market depth tick data (November 2012) collected from the Thomson Reuters Tick History (TRTH) database, for a universe of 42 stocks that make up the South African headline index (TOP40) as at 30 November 2012. This includes 5 levels of order book depth (bid/ask prices and volumes) at each tick. The raw data was imported into a MongoDB database and sampled at regular 5-minute intervals showing prevailing level prices and volumes. The investigation period coincides with the study in Chapter 6, and we will use the appropriate time-scale SSVs in our study below.

8.7.2 Results

We compare two variants of the *Q-learning* implementation: one with *spread* and *quote volume* as public state attributes at resolution 10 (as in Chapter 5), and one which just uses the SSV as the public state attribute, evaluated using different learning rates. *Elapsed time* and *remaining inventory* were used as private state attributes in both models. For both models, the *Q-matrix* was trained using the batch *Q-learning* algorithm shown in Algorithm 3 based on tuples extracted from the period 01 November 2012 to 15 November 2012. The trained *Q-matrix* was then used to select actions in the *test set* (16 November 2012 to 30 November 2012), where 60-minute trading programs were executed (using 5-minute intervals) each hour from 09:00 to 16:00 and resulting IS recorded. In Table 8.2 and Figure 8.1, we show the resulting *difference* in *ex-post* median IS (RL - AC) for each trading time, thus a positive value indicates the candidate model outperformed the static-trajectory AC model.

We see that the best performing model overall is the *spread, volume* model with a learning rate of 0.5. This model is able to achieve an average improvement in median IS of 149 bps across each of the trading start times. The best performing *SSV* model is one with a learning rate of 0.7, achieving an average improvement in IS of 36 bps. It is interesting that the *spread, volume* model offers superior performance to the *SSV* model, however we note that for this interaction scale (5-minute calendar intervals), the *general* SSVs shown in Figure 6.11 were used as the public state attribute. This does not take into account the specific velocity of activity for the stock considered (AGL). We conjecture that *event-time* temporal states, as discussed in Section 6.10 would yield superior results and should be considered in future research. This preliminary study still, however, demonstrates that the *SSV* can be used as a suitable public state attribute to learn an effective policy for optimal trade execution.

Model		Trading Time (H)								Avg
Public attribute	*Learning rate*	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	
Spread,Volume	0.1	0.0012	-0.0008	0.0004	0.0006	0.0123	0.0001	0.0231	-0.0001	0.0055
Spread,Volume	0.2	0.0012	-0.0007	0.0004	0.0006	-0.0008	0.0280	0.0529	0.0228	0.0131
Spread,Volume	0.3	0.0012	-0.0007	0.0004	0.0006	-0.0008	0.0189	0.0536	0.0169	0.0113
Spread,Volume	0.4	0.0012	-0.0006	0.0004	0.0006	0.0241	0.0268	0.0536	0.0120	0.0147
Spread,Volume	0.5	0.0012	-0.0007	0.0004	0.0006	0.0230	0.0268	0.0508	0.0175	0.0149
Spread,Volume	0.6	0.0011	-0.0007	0.0280	0.0005	0.0104	0.0000	0.0415	0.0176	0.0123
Spread,Volume	0.7	0.0011	-0.0006	0.0281	0.0006	-0.0008	0.0000	0.0172	0.0176	0.0079
Spread,Volume	0.8	0.0012	-0.0007	0.0004	0.0005	-0.0008	0.0294	0.0544	0.0081	0.0116
Spread,Volume	0.9	0.0275	-0.0006	0.0004	0.0005	-0.0008	0.0284	0.0206	0.0186	0.0118
Spread,Volume	1.0	0.0120	0.0084	0.0161	0.0005	0.0252	0.0269	0.0286	-0.0002	0.0147
Spread,Volume	var	0.0012	-0.0008	0.0004	0.0006	0.0272	0.0001	0.0076	0.0174	0.0067
SSV	0.1	-0.1267	-0.0270	-0.1339	-0.1338	-0.0144	-0.0130	-0.0044	-0.2286	-0.0827
SSV	0.2	-0.0164	-0.0270	0.0006	0.0009	-0.0074	0.0000	-0.0006	-0.0001	-0.0063
SSV	0.3	0.0011	-0.0008	0.0007	-0.0029	-0.0010	-0.0001	0.0284	-0.0001	0.0032
SSV	0.4	0.0011	-0.0007	0.0007	-0.0029	-0.0009	-0.0001	0.0284	-0.0001	0.0032
SSV	0.5	-0.0013	-0.0008	0.0007	-0.0029	-0.0009	0.0000	0.0284	-0.0001	0.0029
SSV	0.6	0.0013	-0.0008	0.0008	0.0004	-0.0009	0.0000	0.0281	-0.0001	0.0036
SSV	0.7	0.0013	-0.0009	0.0008	0.0004	-0.0008	0.0001	0.0282	-0.0002	0.0036
SSV	0.8	0.0012	-0.0009	0.0007	0.0004	-0.0008	0.0001	0.0282	-0.0002	0.0036
SSV	0.9	0.0013	-0.0008	0.0008	0.0004	-0.0008	0.0001	-0.0008	-0.0002	0.0000
SSV	1.0	0.0012	-0.0007	0.0008	0.0006	-0.0007	0.0001	-0.0008	-0.0002	0.0000
SSV	var	0.0227	-0.0358	0.0004	-0.0069	-0.0136	0.0000	0.0295	-0.0002	-0.0005

TABLE 8.2: Difference (RL - AC) in median *implementation shortfall* for various learning rates. The best *spread, volume* model is highlighted in green and the best *SSV* model is highlighted in red.

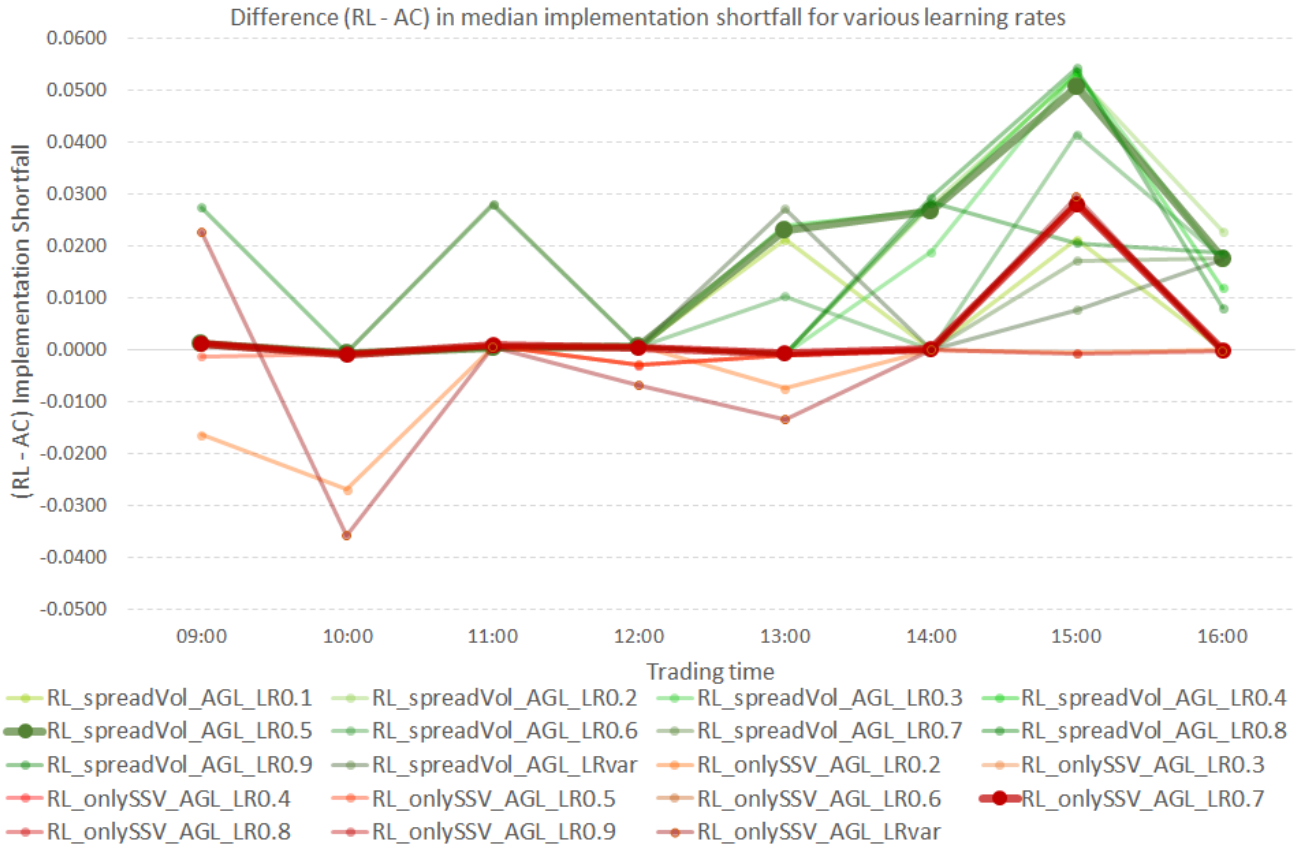


FIGURE 8.1: Difference (RL - AC) in median *implementation shortfall* for various learning rates. The best *spread, volume* model is highlighted by the thick green line and the best *SSV* model is highlighted by the thick red line.

8.8 Some remarks

In this chapter, we demonstrated how the *Q-learning* algorithm for optimal trade execution discussed in Chapter 5 can be modified to incorporate the scale-specific public state attribute for temporal system evolution discussed in Chapter 6. Although more extensive studies are required to verify this technique, the preliminary results suggest that using *SSVs* as the public attribute in the state space can yield effective optimal execution policies, although for the 5-minute calendar time scale considered, the model with stock-specific *spread* and *quote volume* state attributes still offer superior performance. We conjecture that *event-time* temporal states which take into account the velocity of activity for the stock considered, as discussed in Section 6.10, would yield superior results and should be considered in future research. We also note the evolution of the public state attribute is treated as exogenous to the trader's activity, which appears to be a reasonable assumption based on the resiliency study in Chapter 7. In the next chapter, we introduce a technique which relaxes this assumption, where the enumeration of the agent's state space is directly related to what it sees after it's interactions.

Chapter 9

Towards unsupervised, online state discovery, detection and learning in high-frequency financial markets

9.1 Overview

This chapter introduces a scheme for online, unsupervised state discovery, detection and learning in high frequency markets, removing the need for human specification and pre-processing of state attributes, allowing the learning agent to find persistent structure in a streaming market data feed, enumerate its state space and learn to act optimally. It further builds on the premise of the financial market as a complex adaptive system, using this to inform state space discovery and allow adaptation as new niches arise. Chapter 6 provided a scheme for unsupervised, offline estimation of scale-specific temporal states, using the associated SSVs as a public state attribute for state detection. This, however, assumes exogenous evolution of the LOB state space, unaffected by agent interactions. While this may be reasonable at the calendar time scales considered, as the agent interacts at higher frequency time scales, this assumption may be invalidated. The scheme presented in this chapter allows the enumeration of the perceived state space to incorporate the effects of the agent's interactions, via the market data feed it receives, which is more conducive to the event-time trading paradigm.

The contribution is captured in the following paper:

D. Hendricks. *Using real-time cluster configurations of streaming asynchronous features as online state descriptors in financial markets.* Working paper (submitted to Pattern Recognition Letters, under review), 2016. [122]

9.2 Representation learning for tractable inference in high-dimensional state spaces

Machine learning has become ubiquitous in high-frequency financial markets, as technological advances enable low-latency automated algorithms to replace functions traditionally performed by human traders, portfolio managers, risk managers and regulators. This is particularly true for trading algorithms, where reinforcement learning algorithms have recently been considered as dynamic alternatives to traditional stochastic control techniques (such as those found in [13, 35, 51, 89, 91]) for mapping optimal trajectories through the system. Nevmyvaka et al. were among the first authors to consider a reinforcement learning agent for optimal limit order placement for a liquidation program [191, 192]. They used a discrete-state, discrete-action *Q-learning* agent which converged to a policy for the optimal price at which to place the remaining inventory, based on the time remaining in the liquidation program, remaining inventory to trade and domain-informed public state attributes, such as prevailing spreads, price levels and volumes. In Chapter 5 considered a similar problem, demonstrating that a reinforcement learning agent can be used to adapt a static liquidation trajectory with respect to prevailing spread and volume dynamics, executing a sequence of optimised market orders [124]. Both studies demonstrate a significant positive improvement on cost of trading compared to state-of-the-art techniques, motivating reinforcement learning as a suitable framework for online learning agents in financial markets. We do, however, use a subjective set of attributes for state representation in the learning algorithm. While the choice is informed by domain knowledge and may be suitable at the operating scale of a human trader, we conjecture that a more objective representation may yield better trading policies for agents operating at machine scale.

It is well known that the performance of certain classes of machine learning algorithms is strongly dependent on the choice of data representation, or features, upon which they are applied ([34, 127, 162]). This is likely due to certain forms of representation masking exploitable characteristics explaining variations in the data, or at least burying them in layers which cannot be detected by the learning algorithm. As such, significant effort can be spent on data pre-processing, using domain knowledge to inform appropriate representations for effective machine learning. While such human intervention can be useful

to guide learning agents in new domains, it does restrict the agent's discoverable policies to those which mimic policies acceptable to an intuitive *human* agent in the domain. Bengio et al. state that an artificial intelligence should fundamentally understand the world around us, and thus be able to identify and disentangle explanatory features from low-level sensory data without human intervention [34]. In this way, a machine learning agent can provide more general, and sometimes complementary optimal policies to those expected by human agents, thereby cultivating its own distinct *machine intelligence*.

This goal has been recognised by the machine learning community, with a recent surge in scientific activity concerning unsupervised feature learning (or deep learning), seeking the discovery of useful representations which result in more meaningful classifiers and predictors in various domains (see [45, 57, 63, 106, 128, 153] for some state-of-the-art examples). At a recent NIPS workshop, Mnih et al. presented the first deep learning model to successfully learn control policies from high dimensional sensory data using reinforcement learning [185]. The agent was able to learn to play several Atari2600 games, using a convolutional neural network (CNN) trained using a *Q-learning* algorithm, with only raw pixels as the input. While this is a somewhat different domain to the optimal trade execution problem, it does present certain analogues consistent with our goal: using low-level sensory data (*pixels* here, vs *streaming tick data* for our problem), the CNN is able to abstract useful representations from the raw data and train a *Q-learning* agent to achieve some goal. While it would seem appropriate to apply this technique to our problem, the computational burden of the CNN may be too onerous for our goal of an online near-real-time algorithm. Even recent work on state-of-the-art, computationally efficient, GPU-optimised CNNs yield computation times of the order of minutes for relatively modest problems [56].

We are thus tasked with developing a form of state representation which can be constructed directly from raw asynchronous tick data, is able to capture salient features of the limit order book, is computationally efficient for near-real-time use (of the order of seconds) and can be successfully combined with *Q-learning* for optimal trading policies. In the following sections, we describe our approach which is able to construct a rich state representation in < 2 seconds, using relatively modest hardware, enabling near-real-time state detection for online learning.

9.3 Cluster configurations as temporal state descriptors

In the previous studies considering state representations for high-frequency financial markets (and in Chapter 5 here), the following pre-processed attributes were used as candidate descriptors: *bid/ask spread, quote volumes, quote volume imbalance, trade*

price levels and *traded volumes* [124, 191, 192]. These were informed by common notions and intuition from human traders. Given the objective of minimising trading cost with respect to an arrival price benchmark, and the constraint of only executing market orders, a rational trader will attempt to plan his execution trajectory such that he crosses the spread infrequently and with minimal magnitude. Thus, *spread* and *quote volume* were natural candidates for public state attributes if we wish our RL agent to learn this behaviour. In particular, when *spreads* are *tight (wide)* and *volumes* are *high (low)*, we expect the RL agent to trade *more (less) aggressively* to minimise the trading program's overall implementation shortfall.

While informed by domain knowledge and consistent with the data-preprocessing paradigm described by Bengio et al. [34], these choices are somewhat subjective and are only capable of a partial representation of the true state space. Indeed, even the enumeration of all possible *spread* and *volume* configurations at the finest resolution is unlikely to be able to capture evolution and persistency characteristics of the financial system at this scale. While we can increase the complexity of the state space representation by increasing the number of attributes, the *curse of dimensionality* soon prevents computational tractability for an online algorithm, at least in the *Q-learning* setting we consider. In Chapter 6, we introduced a scheme which efficiently summarised persistent information in a multi-featured market data feed into a scale-specific market state attribute describing temporal dynamics. This was added as a public attribute to the agent's state space, and its efficacy investigated in Chapter 8.

We now propose an alternative notion to characterise the state at each decision point, effectively reducing the set of public attributes to a single metric, while preserving information from *all* measurable aspects of the system.

One can think of a particular realisation of state attributes as a cluster configuration of observable features for a stock. Consider the case of the model used in Chapter 5 [124], where *spread* and *quote volume* were used as public attributes. These are derived from the following low-level features of the limit order book: *Level-1 Bid Price, Level-1 Bid Volume, Level-1 Ask Price, Level-1 Ask Volume*. Figure 9.1 illustrates how a cluster configuration of these low-level features has an analogous interpretation to the *low/high spread, low/high volume* regimes described in Chapter 5.

In time period t_1, we see *Level-1 Ask Volume, Level-1 Bid Volume* and *Level-1 Bid Price* are all correlated and increasing, thus being ascribed to the same cluster. *Level-1 Ask Price* is decreasing and is ascribed to another cluster. In particular, we notice that *Level-1 Bid Price* is *increasing* and *Level-1 Ask Price* is *decreasing*, which is consistent with a *narrowing spread* regime. Since we are considering market orders for a *BUY* trading program, we note that the narrow spread is accompanied by a larger *Level-1*

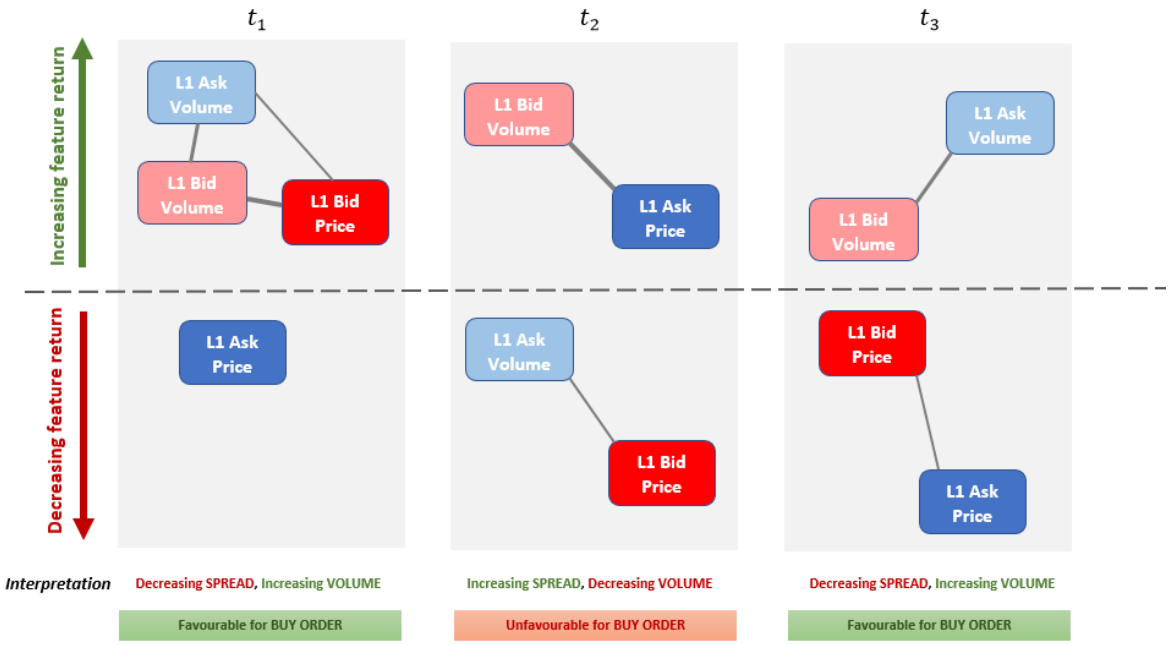


FIGURE 9.1: Illustrating how identified feature configurations may have an analogous interpretation, in terms of human-specified pre-processed features.

Ask Volume, which presents favourable conditions for an increase in trading activity. Thus the *low spread, high quote volume* regime considered in the *SSRQ* model has an analogous feature cluster configuration interpretation.

As a further example, consider the cluster configuration in time period t_2. Here, *Level-1 Ask Price* is increasing, while *Level-1 Bid Price* and *Level-1 Ask Volume* are both decreasing, resulting in a *high spread, low quote volume* regime, consistent with a decrease in trading activity.

This simple illustration demonstrates that the cluster configurations of low-level sensory features in high-frequency financial markets may have an analogous interpretation to the trader-intuition-derived regimes usually specified. Furthermore, by allowing the clustering algorithm to be exposed to streaming data from *all* measurable features, unique and persistent cluster configurations may yield meaningful state representations for machine learning classifiers and predictors, beyond those which may have been expected and proposed by human traders. In addition, an appropriate cluster configuration similarity metric can be used to identify temporal states which are characterised by the same feature cluster configurations. If certain configurations persist throughout the trading day, we then have a reduced number of states for which to solve our optimal trading policy. Thus, we will investigate using cluster configurations to describe temporal regimes in a reinforcement learning framework, utilising a high-speed cluster detection algorithm appropriate for online learning.

9.4 Correlation estimation from streaming asynchronous data

A key input in the clustering algorithm used in this analysis is an object *correlation matrix*.

Classical estimators for co-volatility, and hence correlation, typically rely on evenly-spaced, synchronous observations for computation. In the case of high-frequency data in financial markets, the asynchronous arrival of the price time series would require the use of pre-processing or interpolation techniques before classical techniques can be applied, potentially introducing bias into the results (see [110] for a survey of candidate estimators). Malliavin and Mancino introduced a non-parametric co-volatility estimator based on Fourier series analysis, principally relying on the integration of a time series, rather than its differentiation [170, 171]. Their method removes the need for any artificially-imposed synchronicity, providing a measure of co-volatility which exploits more information in the data, unaffected by data arrival time and sampling frequency.

Using the assumption that asset prices are continuous semi-martingales, Malliavin and Mancino prove a general identity which relates the Fourier transform of the co-volatility function with the Fourier transform of the log price returns [170, 171]. We refer the reader to their papers for the full exposition and proofs. For our purposes, we have made use of their *integrated co-volatility estimator*, defined in Equation 9.1.

$$\hat{\Sigma}^{12}_{n,N} := \frac{1}{2N+1} \sum_{|s|<N} \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} e^{(is(t_i^1 - t_j^2))} \delta_{I_i^1}(p^1) \delta_{J_j^2}(p^2). \tag{9.1}$$

where N is the number of Fourier coefficients, n_k is the number of price changes for asset k in the integrated window, t_i^k is the time of each price change for asset k, $\delta_{I_i^k}(p^k)$ is the consecutive change in log-price of asset k between time t_i^k and t_{i+1}^k. To compute the required pairwise correlation, we compute the volatility for each asset, as well as the co-volatility for both, then use the simple relation in Equation 9.2.

$$\rho^{12} = \frac{(\hat{\Sigma}^{12}_{n,N})^2}{\hat{\Sigma}^{11}_{n,N} \times \hat{\Sigma}^{22}_{n,N}}. \tag{9.2}$$

We note that the use of this estimator for stock co-volatility estimation in the context of the Johannesburg Stock Exchange was first investigated by Malherbe [169]. We will apply this technique for finding *feature* correlations, i.e. considering *features* as *assets* in the exposition above. For the implementation, we used a combination of vectorisation and graphics processing unit (GPU) programming techniques to achieve efficient

computation via parallelisation in MATLAB. Computation of complex exponential coefficients and multiplication with log price differences were performed on the GPU, after which the results in the *gpuArrays* were passed back to the CPU for final computation of co-volatility and correlation. Details of the implementation can be found in Wilcox et al. [247].

9.5 High-speed feature clustering

In Chapter 6, a master-slave parallel genetic algorithm (PGA) with a bespoke log-likelihood fitness function was used to identify emergent stock clusters, based on correlated price evolutions [125]. We provide an efficient computational solution for a technique initially proposed by Blatt et al. [38, 39] and Giada and Marsili [103, 104], where the super-paramagnetic ordering of a q-state Potts model is used for cluster identification. In a market Potts model, each stock can take on one of q possible states, and each state can be represented by a cluster of similar stocks. Cluster membership is indicative of some commonality among the cluster members. Each stock has a component of its dynamics as a function of the state it is in and a component of its dynamics influenced by stock specific noise. In addition, there may be global couplings that influence all the stocks, i.e. the external field that represents a market mode.

We refer the reader to Chapter 6 for a comprehensive discussion of the technique and derivation of the pertinent log-likelihood fitness function [103, 125, 126]. In particular, from Section 6.4, we note that for a candidate cluster configuration of N objects, $\mathcal{S} = \{s_1, ..., s_N\}$, the log-likelihood of $\mathcal{S}$ explaining the structure inherent in the data is given by

$$\mathcal{L}_c(\mathcal{S}) = \frac{1}{2} \sum_{s:n_s>1} \left(\log \frac{n_s}{c_s} + (n_s - 1) \log \frac{n_s^2 - n_s}{n_s^2 - c_s} \right), \tag{9.3}$$

where

$$n_s = \sum_{i=1}^{N} \delta_{s_i,s} \tag{9.4}$$

is the number of objects in the sth cluster and

$$c_s = \sum_{i=1}^{N} \sum_{j=1}^{N} C_{i,j} \delta_{s_i,s} \delta_{s_j,s} \tag{9.5}$$

is the intra-cluster correlation. In Equation 9.5,

$$C_{i,j} = \frac{\bar{x_i} \bar{x_j}}{\sqrt{\|\bar{x_i}^2\| \|\bar{x_j}^2\|}} \tag{9.6}$$

is the Pearson correlation between the ith and j^{th} objects, representing the short-range distance-dependent ferromagnetic interaction term in the Potts analogy.

In Chapter 6 show that the likelihood function specified in Equation 9.3 can be used as an objective function in a high-speed, scalable parallel genetic algorithm (PGA), where candidate cluster configurations are evaluated and successively improved until a configuration best explains the inherent structure suggested by a correlation matrix [125]. We will utilise this computational solution, since it provides the near-realtime efficiency required for our proposed online algorithm.

It is important to note that this approach, applied to stock features as objects, is consistent with our view of financial markets as complex adaptive systems. Essentially, if we return to the spin glass model analogy described in Chapter 1, we are finding a spin glass configuration (feature clustering) which permits a *metastable* system state given the spin-interaction term (feature correlations).

9.6 Cluster configuration similarity and state discrimination

In Sections 9.4 and 9.5, we demonstrated a scheme to determine the feature cluster configuration from raw asynchronous data streaming from a market data feed, over an integrated window. Given this choice of state representation, what remains is to provide a feasible scheme to discriminate between states, such that the state space can be enumerated online.

This is a special case of a more general problem: measuring the distance between overlapping cluster configurations of a fixed set of objects. Consider two candidate cluster configurations of a fixed set of n objects, $C_1 = \{s_1, s_2, ..., s_n\}$ and $C_2 = \{s'_1, s'_2, ..., s'_n\}$, where s_k and s'_k are the cluster indices to which the kth object belongs in each configuration. We would like to define a distance metric $d(C_1, C_2)$ which quantifies the configuration differences, while preserving the properties of symmetry, identity of indiscernables, non-negativity and sub-additivity.

Goldberg et al. considered this problem and proposed three candidate measures to quantify cluster configuration differences [108]. We chose to implement a variant on their *best match* metric, which effectively counts the number of *moves* required to convert one configuration to the other. The measure is defined as,

$$d(C_1, C_2) = \frac{n}{n^2 - 1} \sum_{i=1}^{n} \min_j d(s_i, s'_j), \tag{9.7}$$

where $d(s, s') = 1 - \frac{|s \cap s'|}{|s \cup s'|}$ and $\frac{n}{n^2-1}$ is a normalisation constant, such that the maximum distance between two configurations is 1 (*single cluster* vs *all singletons*). For example, consider the following two candidate cluster configurations:

$$C_1 = \{1, 2, 3, 3, 4, 4, 4, 5\}$$
$$C_2 = \{1, 2, 2, 2, 3, 3, 4, 5\}.$$

Then, when considering cluster index 2, $d(s_2, s'_2) = 1 - \frac{|s_2 \cap s'_2|}{|s_2 \cup s'_2|} = 1 - \frac{1}{3} = 0.67$. Table 9.1 illustrates the full calculation of the distance between configurations C_1 and C_2.

Cluster index k	$\lvert s_k \cap s'_k \rvert$	$\lvert s_k \cup s'_k \rvert$	$d(s_k, s'_k)$
1	1	1	0.00
2	1	3	0.67
3	0	4	1.00
4	1	3	0.67
5	1	1	0.00
$d(C_1, C_2)$			$\frac{2.34}{4.8} = \mathbf{0.49}$

TABLE 9.1: Demonstration of *best match* metric for calculating distance between two overlapping cluster configurations

Given a quantified distance between cluster configurations, we need to specify some *distance threshold* which encodes the idea that the configurations are sufficiently similar to be categorised as the same state. The specification of this *distance threshold* is somewhat ad-hoc and a source for subjective input, however coupled with a learning objective, we can iterate through multiple candidate thresholds to determine one which optimises the objective. In the use-case demonstrated in Section 9.9, we consider a simple *Q-learning* algorithm which aims to maximise wealth by deciding when to buy/sell shares. Starting with a given cash amount and stock inventory, we provide the learning agent with a fixed set of actions (buy/sell volumes) which it can perform in each period, based on the prevailing state. Each candidate distance threshold will provide the agent with a different lens to discriminate states, hence the state-action policy and terminal wealth of the agent will be impacted. We will choose a threshold which maximises terminal wealth.

The *best match* metric is easy to compute, intuitive and efficient, however the alternative metrics proposed by Goldberg et al. [108] should be explored in further research, to assess the impact on state discrimination.

9.7 Reinforcement learning with online state discovery

In Chapter 3, we discussed the RL technique for finding a calibrated policy in a controlled Markovian system with unknown dynamics, mapping system states to optimal or near-optimal decisions, given some objective. Learning can be model-free ([243]) or model-based ([229]), however the key principle is that feedbacks from interactions with the system can be used to provide insight for optimal planning decisions. In Chapters 5 and 8, we focused on model-free Q-learning, and we refer the reader to [147] for a comprehensive review of RL techniques.

We will consider using the state space enumeration technique described in Sections 9.3 to 9.6 for an RL agent, enabling online planning decisions to be made with an adaptive state space.

The learning algorithm we propose here can be seen as a particular implementation of the *Dyna-Q* architecture proposed by [229, 230], whereby feedbacks from the system are simultaneously used to improve the model of the system dynamics (state transitions), as well as the state-action policy for the learning objective. Both the feedbacks (or rewards) and expected state transitions are used to enumerate a so-called *Q-matrix* online, which contains the (current) discounted expected reward for each state-action pair, assuming the optimal policy is followed after the current time-step [243]. At the tth step in the learning algorithm, the agent:

We recall that in the tth episode, the agent:

- observes its current state $S_t \in \mathcal{S}$,
- selects and performs an action $A_t \in \mathcal{A}$,
- observes the subsequent state S_{t+1} as a result of performing action A_t,
- receives an immediate reward r_t and
- uses a learning factor α_t, which decreases gradually over time.

Q is updated as follows:

$$Q_{t+1}(S_t, A_t) \quad = \quad Q_t(S_t, A_t) + \alpha_t[r_t + \gamma \max_b Q_t(S_{t+1}, b) - Q_t(S_t, A_t)], \tag{9.8}$$

where γ is the *discount rate* controlling the importance of future rewards. In our prior specification, the *state space* $\mathcal{S}$ and permissible actions $\mathcal{A}$ were fixed *a priori*. We now consider the case where the action set $\mathcal{A}$ remains fixed, but the state space $\mathcal{S}$ is dynamic, viz. discovered online as the agent interacts with the system. Each time a new state

is *discovered* using the proposition in Section 9.6, the state space is increased. For the update rule in Equation 9.8, the *next state* is determined using the prevailing empirical transition probability matrix, as

$$S_{t+1} = \arg\max_{S_j} Pr(S_t, S_j) \text{ for } S_j \in \mathcal{S}.$$

For the purposes of the investigation in Section 9.9, we assume that the agent's actions do not affect the system state evolution. This is an artefact of interacting with a *historical* data feed, where the consequences of an agent's actions cannot easily be incorporated. The overall proposition is, however, designed for a live trading agent submitting actual market orders, thus a *live* trading agent will affect the data feed it receives (absorbing limit orders through trades, affecting the LOB features), and thus the state space it perceives. We expect the efficacy demonstrated in Section 9.9 to translate to live trading.

9.8 Problem description and Algorithm

9.8.1 Wealth maximisation: *Long-only*

To test the efficacy of the framework proposed in this chapter, we construct a wealth maximising trading agent operating in high-frequency financial markets, able to buy and sell quantities of *one* given stock. The agent begins with a specified level of *cash* and *stock inventory*. At each trading opportunity (we assume regular 5-minute periods, however this can be generalised), the agent is able to *buy* certain quantities of the stock using available *cash*, or sell certain quantities of stock based on the level of *inventory*. We assume the agent is subject to a *long-only* constraint, i.e. the agent is not able to short-sell inventory, and is not allowed to use leverage. The *reward* is calculated as the portfolio PnL following the chosen action, i.e. the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. The price associated with all *buy* actions is the prevailing *best ask price*, and for all *sell* actions the prevailing *best bid price*. This ensures that *spread* is included as a transaction cost when trading. The agent uses the asynchronous tick-level data observed over the preceeding 5-minute window to compute feature correlations and the associated cluster configuration, as described in Sections 9.4 and 9.5, before determining the state, using the proposition in Section 9.6, and updating the *Q-matrix*.

Table 9.2 shows the parameters which were used in the results which follow.

Stock	AGL
Features	Trade Price/Volume, L1 Ask/Bid Price/Volume
Initial Cash	R100 000
Initial Inventory	800 shares
Buy Actions (proportion of Cash)	$\{0, 0.1, 0.2, ..., 0.9, 1.0\}$
Sell Actions (proportion of Inventory)	$\{0.1, 0.2, 0.3, ..., 0.9, 1.0\}$
State discrimination threshold	0.05
Probability of random action	0.05
Start time	09:05
End time	16:30
Estimation period / Trading frequency	5 minutes
Initialisation period	5 periods

TABLE 9.2: Parameters used for testing long-only wealth maximisation algorithm.

9.8.2 Algorithm

Algorithm 4 describes the general implementation of the learning algorithm with online state discovery. We begin by initialising an empty *Q-matrix*, as no states have been discovered. After the first estimation period has passed (in this case 5 minutes), the raw feature data in the estimation period window is used to compute feature correlations, and then the feature cluster configuration. We then compute the distance between the current feature configuration and the previously-identified configurations associated with prevailing states in the state space. If the distance between the current configuration and a previously-identified configuration is less than the specified threshold, we ascribe the associated state index to the current configuration. If the distance to all previously-identified configurations is larger than the threshold, a new state is created. The prevailing transition probability matrix is updated with the new state. We then use the ϵ-*greedy* algorithm to choose an action based on the prevailing *Q-matrix*, recording the reward as the difference between the marked-to-market portfolio value and initial portfolio value. The transition probability matrix is then used to identify the next expected state, and the *Q-value* associated with the current state-action pair is updated using Equation 9.8.

Algorithm 4 Unsupervised state detection and learning

Initialise *Q-matrix*
while Trading program not complete **do**
  Extract feature time-series (raw, asynchronous events) for integrated window
  Compute feature correlations using Fourier estimator
  Compute feature cluster configuration $\mathcal{C}$ using PGA
  Compare $\mathcal{C}$ with previously identified configurations $\mathcal{C}_i \in$ ***state space***
  Update state space
  if ***state space*** $= \emptyset$ **then**
    Generate new state
  else if distance($\mathcal{C}$,$\mathcal{C}_i$) $\leq$ *threshold* for any $\mathcal{C}_i \in$ ***state space*** **then**
    Assign state index of $\mathcal{C}_i$ to $\mathcal{C}$
  else
    Generate new state
  end if
  Update state transition probability matrix
  Update empirical prob of 1-step transition given all identified states
  if initialisation period complete **then**
    Choose current optimal buy/sell action using Q-matrix
    Record reward R as difference between current MTM portfolio value
    and initial portfolio value
  end if
  Update Q-matrix
  Determine next state using current transition prob matrix
  Update Q-value associated with state-action pair, given recorded reward, next state
  and prevailing Q-value
end while

9.9 Data and Results

9.9.1 Data

The data for this study constituted tick-level trades and top-of-book quotes for one candidate stock on the Johannesburg Stock Exchange (JSE) from 1 October 2012 to 30 November 2012. This data was sourced from the Thomson Reuters Tick History (TRTH) database. The raw data was stored in a MongoDB noSQL database, with appropriate indexes created for efficient retrieval and manipulation. The particular fields of interest for our study are: *Trade Price, Trade Volume, Level-1 Bid Price, Level-1 Bid Volume, Level-1 Ask Price, Level-1 Ask Volume*. Each of these features are represented by an unevenly-spaced time series in the dataset based on event occurrence. The stored, asynchronous event data is a close approximation to a stream of asynchronous events arriving from a live market data feed. We will apply our learning algorithm to this data in its most raw form, to avoid any subjective bias which may be introduced by data pre-processing techniques.

9.9.2 Results

Table 9.3 shows the summarised results from our analysis. We ran the algorithm for each day in our data set (01 October 2012 to 30 November 2012), constructing a distribution of end-of-day (16:30) PnL, expressed as a percentage of initial portfolio value. A positive percentage thus indicates that the algorithm decisions *created value* over the trading day. To test the efficacy of our algorithm, we compared its performance to a *random agent*. The *random agent* chooses actions randomly at each decision point, i.e. makes no use of the learnt *Q-matrix*. This will allow us to test whether our algorithm is at least better than choosing actions at random. The table shows the minimum, lower quartile (LQ), mean, median, upper quartile (UQ), maximum and standard deviation of the end-of-day PnL distributions for both agents.

AGL	**PnL (as % of initial portfolio value)**						
Model	Min	LQ	**Mean**	**Median**	UQ	Max	Std Dev
LO Wealth Maximiser	-2.16	-0.38	**0.12**	**0.18**	0.53	2.77	0.98
Random	-2.62	-1.10	**-0.61**	**-0.48**	-0.24	1.30	0.87
Difference	0.45	0.72	0.73	0.66	0.77	1.47	0.12

TABLE 9.3: Summarised results from algorithm testing. The *LO Wealth Maximiser* agent is compared to a *Random* agent, where actions are chosen randomly at each trading opportunity. The algorithm begins at 09:05 and ends at 16:30 each trading day. These results summarise the distribution of end-of-day (16:30) PnL recorded for each day in the investigation period (01 Oct 2012 to 30 Nov 2012).

Based on the results in Table 9.3, we see that the *LO wealth maximiser* agent generates a significantly better mean and median end-of-day PnL compared to the *random agent*, over this investigation period. In fact, the entire distribution for the *LO wealth maximiser* is more positively skewed, indicating that, in general, the agent generated a modest, but positive daily PnL. This is significant, as after only 5 periods of initialisation to refine the transition probability matrix, with no prior training, the agent is able to learn a useful policy fast enough to generate a positive PnL by the end of the trading day. While many more tests need to be run, varying the parameters in Table 9.2 and considering longer (and varied) investigation periods, these results indicate that the approach suggested in this chapter may be an effective framework for deploying purposeful trading agents which are able to identify exploitable structure in streaming market data feeds, and learn policies fast enough.

Figures 9.2 to 9.6 illustrate a typical run of the online algorithm, at various stages in the trading day, showing the identified states (top-left), prevailing transition probability matrix (top-right), portfolio PnL, stock mid-price, best bid and best ask levels (bottom-left) and current *Q-matrix* values (bottom-right). For the transition probability matrix

and *Q-matrix* values, green indicates higher positive values (darker is higher), red/orange indicates lower values (darker is lower) and grey indicates an uninitialised state transition/state-action pair. In the current PnL plot, green dots indicate *buy* decisions and red dots indicate *sell* decisions, with the size of the dot being proportional to the quantity.

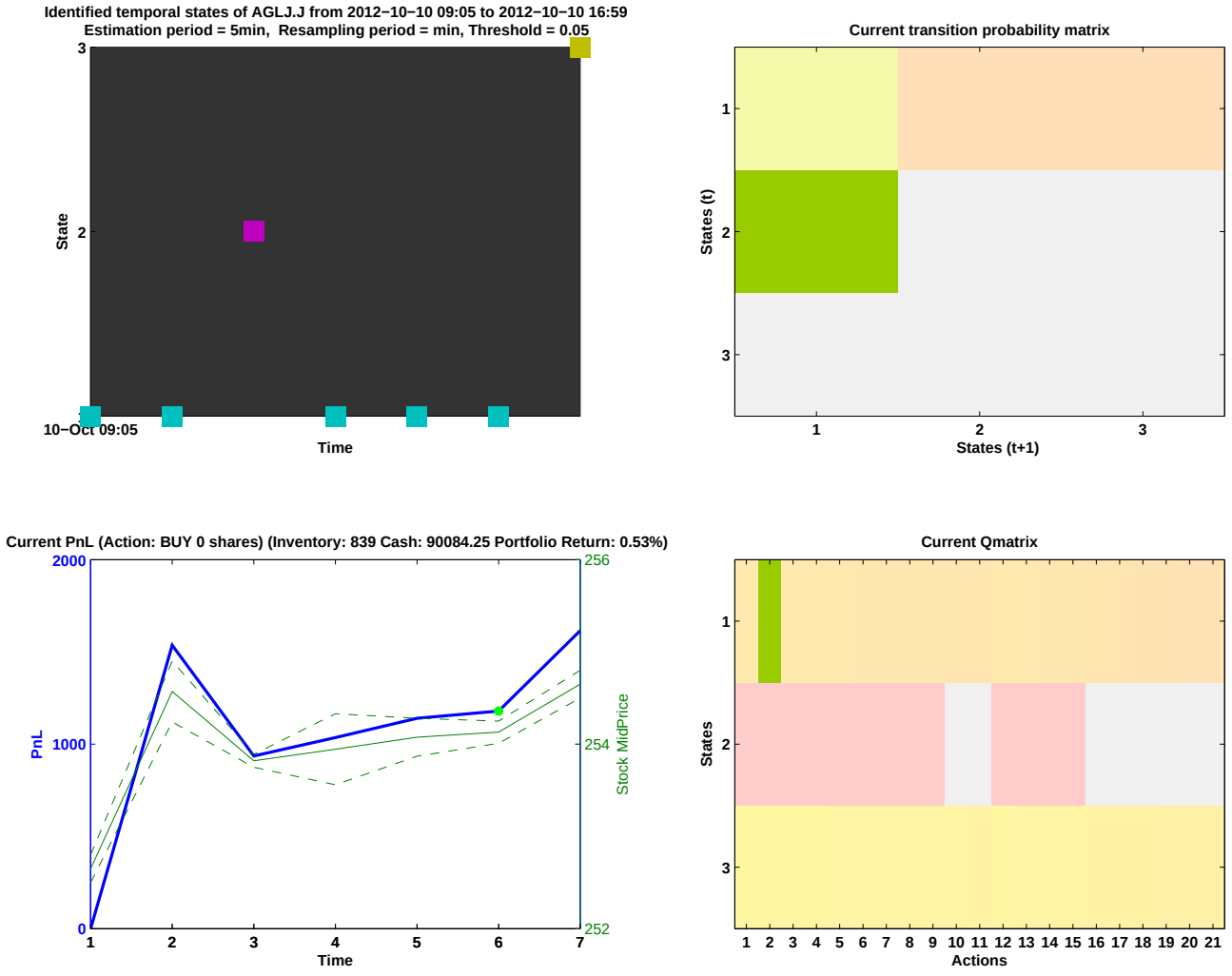


FIGURE 9.2: *Status at 09:35.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair.

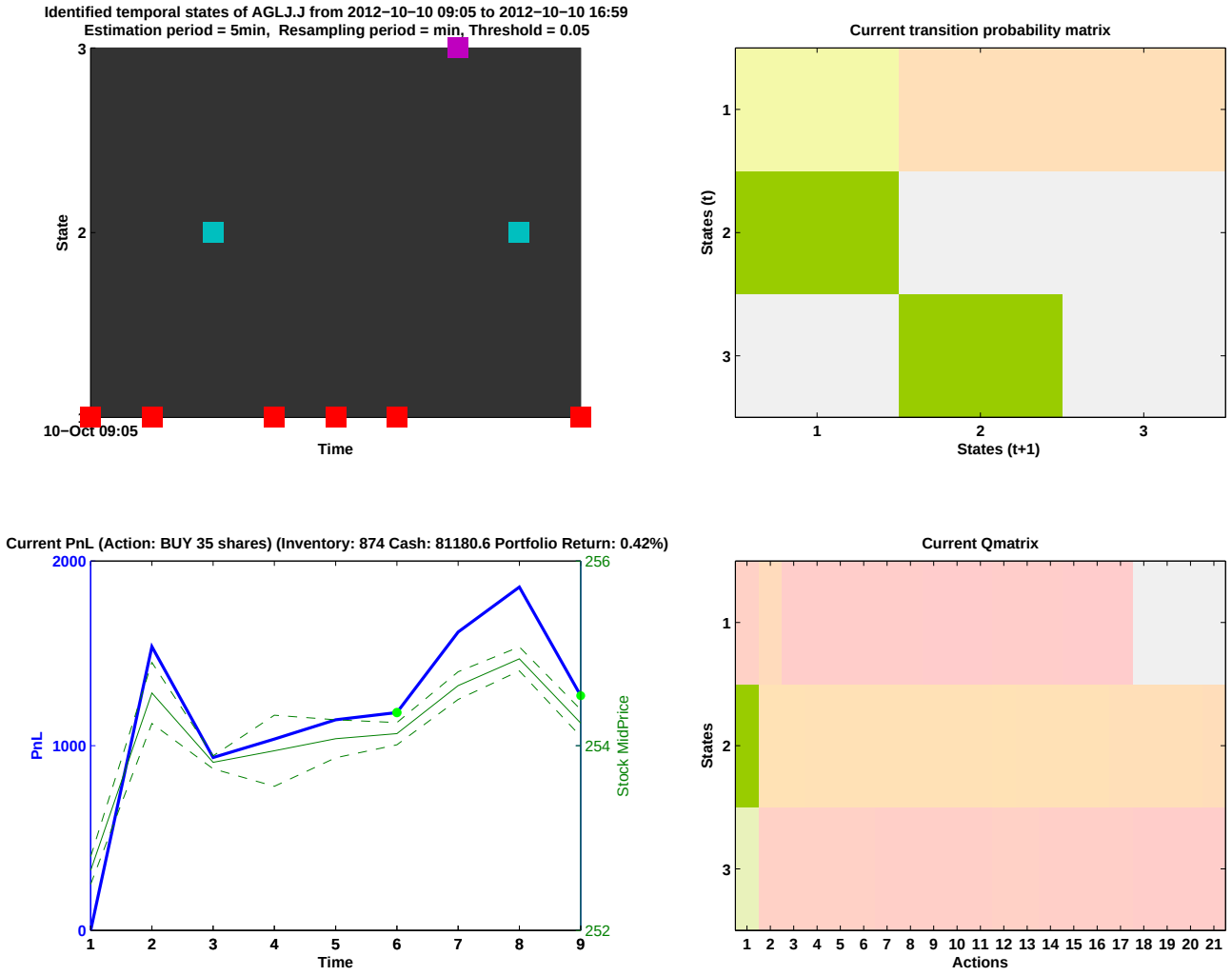


FIGURE 9.3: *Status at 09:45.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair.

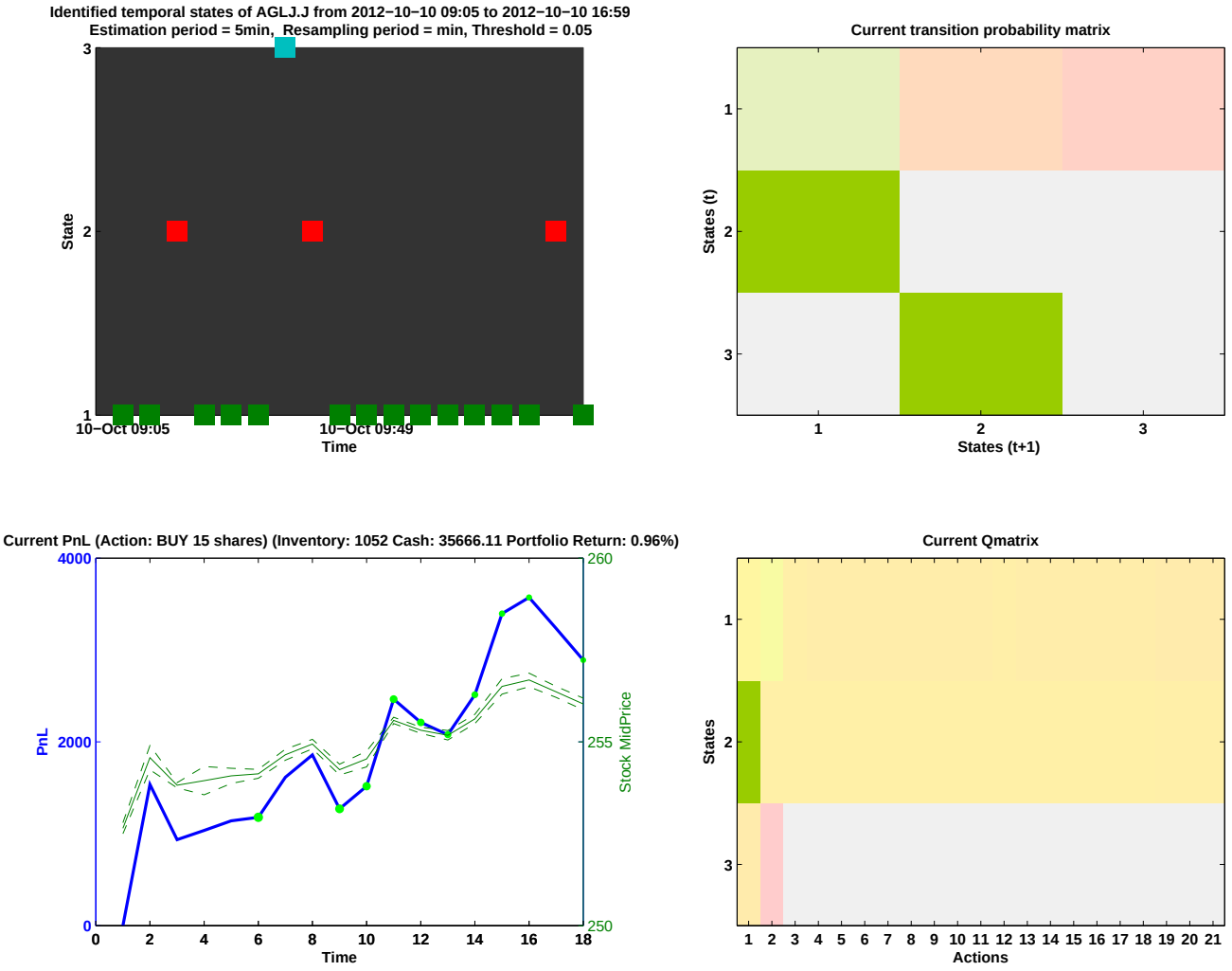


FIGURE 9.4: *Status at 10:30.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair.

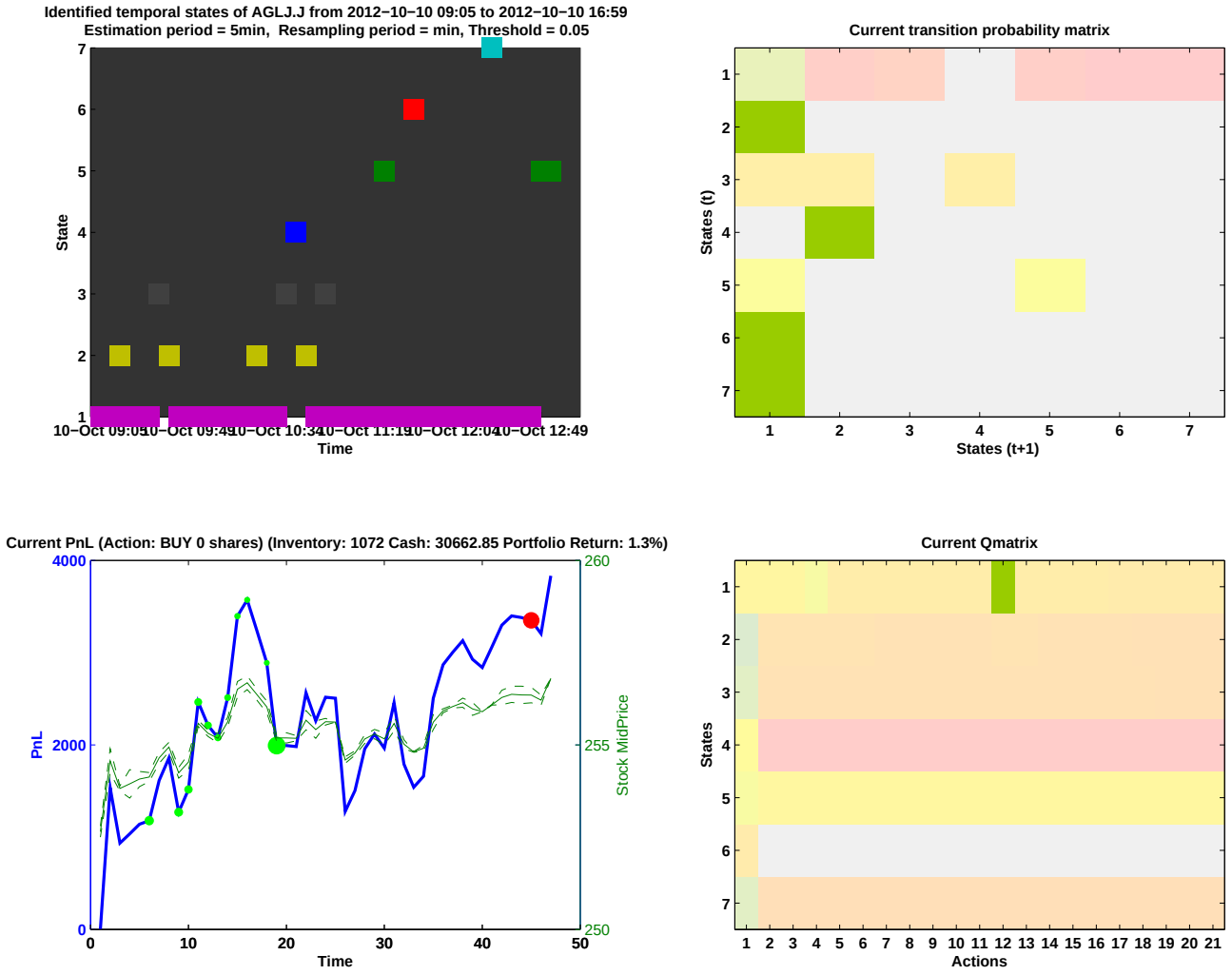


FIGURE 9.5: *Status at 12:45.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair.

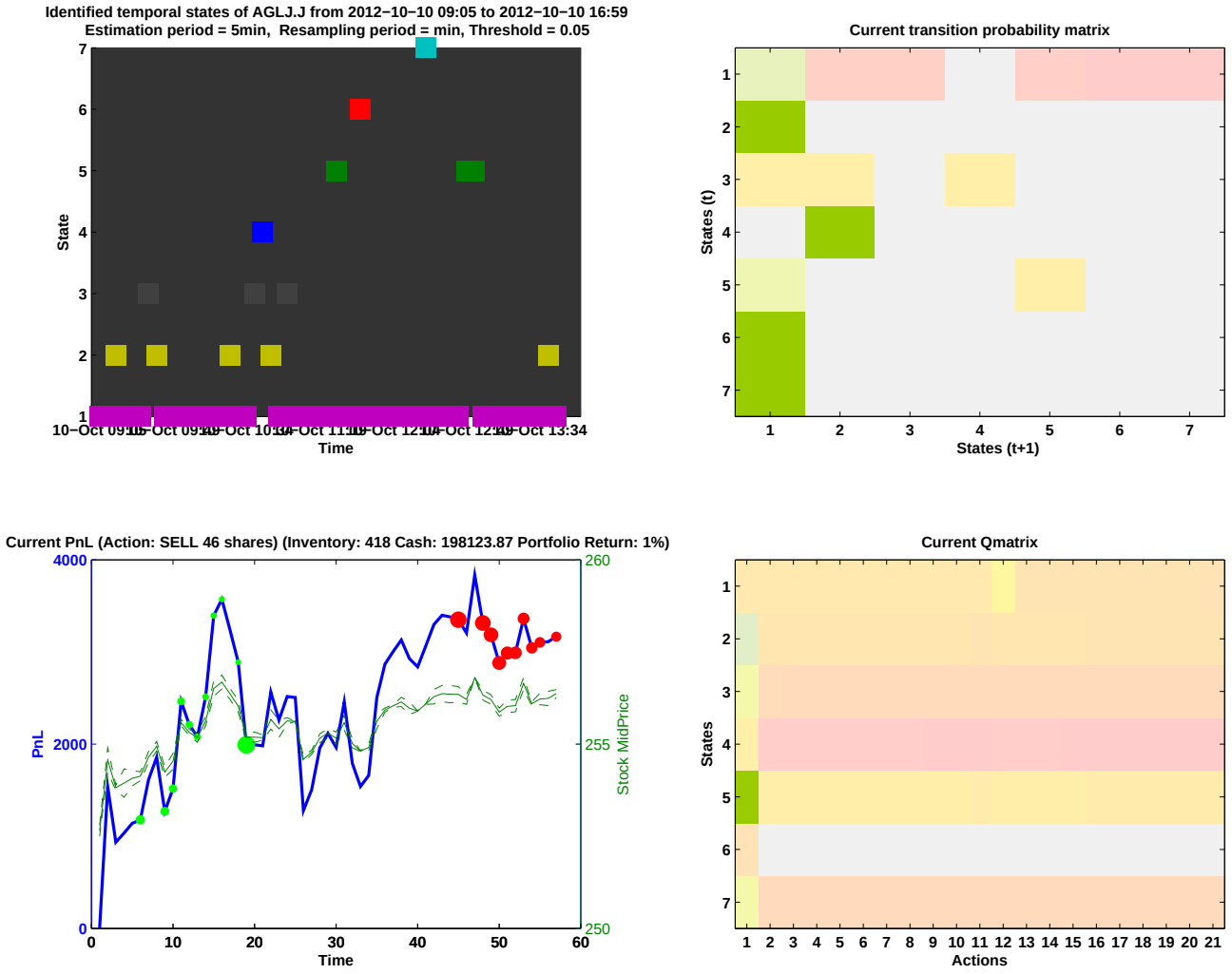


FIGURE 9.6: *Status at 13:30.* Demonstration of long-only wealth maximisation algorithm, starting with R100 000 cash and 800 AGL shares. The *top-left* plot shows the identified states since the start of the trading program (09:05), where blocks in the same row indicate the same state. The *top-right* plot illustrates the current empirical 1-step transition probability matrix, based on identified states. The *bottom-left* plot shows the stock mid-price, best ask and best bid in green (right Y-axis) and the running portfolio PnL in blue (left Y-axis). The portfolio PnL is determined by the difference between the current portfolio value (inventory marked-to-market at current mid-price + cash) and the initial portfolio value. Green dots indicate *buy* actions and red dots indicate *sell* actions, where the size of the dot is proportional to the quantity bought/sold. The *bottom-right* plot shows the current Q-matrix values, illustrating the expected cumulative discounted reward for each state-action pair.

9.10 Some remarks

In this chapter, we demonstrated a scheme for online, unsupervised state discovery, detection and learning in high frequency markets, which is consistent with the complex adaptive system paradigm. By treating stock features as objects and applying the Potts model clustering approach introduced in Chapter 6, we are essentially finding a spin glass configuration (feature clustering) which permits a *metastable* system state given the spin-interaction term (feature correlations). Combined with a candidate state discrimination technique, this allows us to enumerate the state space online, permitting a purposeful agent to learn useful policies in an unknown domain without knowledge of human-preprocessed features and states.

The framework in this chapter is conceptual and there are many areas for refinement, however we demonstrated that even with a simple choice of cluster distance metric and only top-of-book LOB features, a wealth-maximising agent can systematically outperform a random agent over the investigation period considered. In particular, after only 5 periods of initialisation to refine the transition probability matrix, with no prior training, the agent is able to learn a useful policy fast enough to generate a positive PnL by the end of the trading day. This is a promising result for the technique and bodes well for further research.

We note that we have assumed that the agent's actions do not affect the system state evolution. This is an artefact of interacting with a *historical* data feed, where the consequences of an agent's actions cannot easily be incorporated. The overall proposition is, however, designed for a live trading agent submitting actual market orders, thus a *live* trading agent will affect the data feed it receives (absorbing limit orders through trades, affecting the LOB features), and thus the state space it perceives. We expect the efficacy of the algorithm to translate to live trading.

Prudent further investigations would include using alternative streaming features from the LOB (such as market depth), testing the stability of the algorithm over longer and more varied investigation periods, using *event-time* estimation windows / decision frequencies and alternative actions and reward functions.

Chapter 10

Conclusion

In this thesis, we aimed to provide a feasible scheme for automated trading agents to navigate high-frequency financial markets, when viewed as a complex adaptive system. As more functions traditionally performed by human agents in financial markets become replaced by machines, it is critical to appreciate the *lens* of machine agents for observing system dynamics, combining this with a robust paradigm for scale-specific governing dynamics to ensure persistent structure can be identified and exploited.

In Chapter 1, we introduced the key ideas which contextualise the contribution of this thesis. We first motivated the view of the financial market as a complex adaptive system, where agents with varying system visibility collectively and simultaneously create and perceive their environment, acting in service of some goal. The existence of multiple levels of competing, purposeful agents in financial markets, each operating at a different scale with different effective models governing their behaviour allows for the emergence of complex behaviour when viewed in aggregate. In Section 1.2, we motivate the use of *spin glass models* as a tractable formalism to capture the complex nature of objects in the financial system. Recent technological advances, accelerated by a highly competitive industry, have allowed for the efficient generation, storage and retrieval of financial data at high-resolution time scales, providing a rich record of the price formation process as a laboratory for intensive study. This dense, multi-featured, asynchronous data set allows us to study the microstructure of price formation and interrogate system properties, however it introduces computational challenges for efficient calibration.

Chapter 2 discusses the field of market microstructure and highlights key features of our system of interest: the *limit order book*. In particular, we were interested in understanding the rules governing the submission of different order types to the system, the effect of trade events on observed features and any permissible trade-offs relevant for the optimal trade execution problem. This chapter outlines the optimal trade execution

problem with arrival price objective, which serves as the primary problem considered in this thesis. The concepts of *price impact* and *resiliency* are discussed as primary considerations for controlling the agent's interactions with the system, to minimise material impact on LOB dynamics.

Chapter 3 introduces a candidate paradigm for learning optimal state-action policy mappings in this system under Markovian dynamics, viz. *reinforcement learning*. RL promotes a mechanism for online learning of optimal trajectories through a system with unknown dynamics, using feedbacks from interactions with the system for learning. In particular, we consider discrete-action, discrete-state *Q-learning*, which offers guaranteed convergence to the globally optimal policy under certain conditions. We aimed to use the principle of reinforcement learning, i.e. learning from experience, to assess whether it can be used to learn a *useful* policy *fast enough* when interacting with a complex adaptive system at varying scales. As discussed in Section 3.8, the learning rate becomes a crucial parameter and was given special consideration in Chapter 8.

Chapter 4 describes the nature of the raw data used for our investigations, viz. high-frequency trade and quote tick data with 10 levels of market depth, at microsecond resolution. Each event in the dataset has its exact arrival time and date, creating a dense set of asynchronously-arriving events. We uploaded the raw data into a MongoDB noSQL database, which provided efficient storage, retrieval, query indexing and aggregation features for handling large data sets. A bespoke API was written to ensure seamless integration into MATLAB, which was our primary scientific computing environment. This chapter also uses the *exploratory data analysis* paradigm of John Tukey to provide a preliminary visualisation of different aspects of the raw data, highlighting key features which inform the modelling decisions which followed.

Chapter 5 introduces a *Q-learning* algorithm for optimal trade execution, using pre-processed features to enumerate a discrete state space at a specified resolution, with market order volume as the chosen control. The learning algorithm is used to adapt a static liquidation trajectory with respect to prevailing order book features, in order to improve the post-trade implementation shortfall with respect to the program's arrival price. We show that reinforcement learning can be used successfully to modify a given volume trajectory based on market attributes, executed via a sequence of *market orders* based on the prevailing limit order book. Using a sample of stocks and trade sizes in the South African equity market, we were able to reliably improve post-trade *implementation shortfall* by up to 10.3% on average for short trade horizons, demonstrating promising potential applications of this technique.

Chapter 6 considers developing a public state attribute for the state space of the learning agent which is appropriate for the scale of interaction, consistent with the complex

system paradigm, not reliant on arbitrary resolution specification for discretisation, and unaffected by the *curse of dimensionality*. We propose a novel approach for the unsupervised detection of intraday temporal market states at varying time scales, as well as a mechanism for significant state selection and online state detection. By assigning spins to periods as objects and treating period correlations as a short-range interaction term, a q-state Potts model is used to find the configuration of temporal periods which coincides with the system ground state, where spin alignment suggests object homogeneity. Regions of aligned spins are thus treated as clusters or states. We also introduce a high-speed parallel genetic algorithm for finding the best approximation of this structure in a highly efficient and scalable manner, suiting overnight or intraday calibration, or even the online application discussed in Chapter 9. A study of temporal cluster configurations and power-law fits to 60-minute, 30-minute, 15-minute and 5-minute time scales revealed scale-specific system behaviour, motivating the need for scale-specific state space reduction for optimal planning of participating trading agents. The proposed scheme for online state detection suggested the use of SSVs to capture the market activity signature of each identified state, with a simple distance metric of the prevailing FV to determine the state index. We showed that the online state detection scheme can be used to enumerate and update 1-step transition probability matrices, which can be used for optimal planning in the high-frequency trading domain. We considered the stability of the algorithm *ex-post* and found that we could reliably determine 30-minute, 15-minute and 5-minute states using the proposed algorithm, whereas 60-minute states were less stable. A preliminary study of *event-time* clusters reveals its feasibility as an extension of this work, to construct state representations *through the lens of the stock considered*, providing a temporal state evolution appropriate for the trading scale of the agent.

Chapter 7 introduces a scheme to assess the impact of agent interactions on the system. A multivariate Hawkes process is used to measure the resiliency of the limit order book with respect to liquidity-demand events of varying size. By studying the branching ratios associated with key quote replenishment intensities following trades, we can ensure that the limit order book is expected to be resilient with respect to the maximum permissible trade executed by the agent. We find that, for the particular stock and time period considered, a heuristic choice of twice the mean trade volume as an upper bound on trade size should provide a reasonable guarantee of resiliency.

Chapter 8 demonstrates how the *Q-learning* algorithm for optimal trade execution discussed in Chapter 5 can be modified to incorporate the scale-specific public state attribute for temporal system evolution discussed in Chapter 6. We also incorporate the trade size ceiling discussed in Chapter 7. The preliminary results suggest that using *SSVs* as the public attribute in the state space can yield effective optimal execution policies,

although for the 5-minute calendar time scale considered, the model with stock-specific *spread* and *quote volume* state attributes still offer superior performance. We conjecture that *event-time* temporal states which take into account the velocity of activity for the stock considered, as discussed in Section 6.10, would yield superior results and should be considered in future research.

Chapter 9 introduces a scheme for online, unsupervised state discovery, detection and learning in high frequency markets, removing the need for human specification and preprocessing of state attributes, allowing the learning agent to find persistent structure in a streaming market data feed, enumerate its state space and learn to act optimally. While it differs somewhat from the approach considered in the rest of this thesis, it further builds on the premise of the financial market as a complex adaptive system, using this to inform state space discovery and allow adaptation as new niches arise. By treating stock features as objects and applying the Potts model clustering approach introduced in Chapter 6, we are essentially finding a spin glass configuration (feature clustering) which permits a *metastable* system state given the spin-interaction term (feature correlations). Combined with a candidate state discrimination technique, this allows us to enumerate the state space online, permitting a purposeful agent to learn useful policies in an unknown domain without knowledge of human-preprocessed features and states. We demonstrated that even with a simple choice of cluster distance metric and only top-of-book LOB features, a wealth-maximising agent can systematically outperform a random agent over the investigation period considered. In particular, after only 5 periods of initialisation to refine the transition probability matrix, with no prior training, the agent is able to learn a useful policy fast enough to generate a positive PnL by the end of the trading day. This is a promising result for the technique and bodes well for further research. We note that we have assumed that the agent's actions do not affect the system state evolution. This is an artefact of interacting with a *historical* data feed, where the consequences of an agent's actions cannot easily be incorporated. The overall proposition is, however, designed for a live trading agent submitting actual market orders, thus a *live* trading agent will affect the data feed it receives (absorbing limit orders through trades, affecting the LOB features), and thus the state space it perceives. We expect the efficacy of the algorithm to translate to live trading.

This thesis thus contributes two approaches for enumerating the state space of a learning agent in financial markets. The first is an offline, scale-specific study of temporal periods to reveal system states, from which SSVs are extracted which enable online state detection. This assumes the temporal evolution of the system is exogenous to the trader, which is reasonable for trade actions up to a certain size, as demonstrated by the LOB resiliency study. The second approach is to enumerate the state space online, at the scale at which the agent interacts with the system. By construction, the

effects of the live trading agent on LOB dynamics are thus incorporated into the market data feed, and hence the perceived state evolution. We conjecture that this approach is general enough for any purposeful agent to learn a useful policy from a multi-featured, asynchronously-arriving data feed *fast enough* at the scale of interaction, however this needs to be verified. Further studies should focus on the event-time extension of these approaches, where stock features (such as traded volume) govern the velocity of *ticks* in the observed system. This is where the true confluence of the contributions of this thesis lies for developing practical trading algorithms in high-frequency markets: a rigorous understanding of emergent phenomenology at the event scale in market microstructure (Chapters 4 and 7), constructing a state space at the scale (calendar or event) of participation (Chapters 6 and 9), providing a visibility for the trading agent which permits learning a useful *adaptive* execution policy *fast enough* (Chapters 5, 8 and 9).

More broadly, this provides a framework for a dynamic learning algorithm, capable of determining exploitable structure in a system with high throughput of streaming events, learning an optimal policy online for any objective (defined through the reward function) and adapting as market regimes shift. The preliminary positive efficacy of the algorithm proposed in Chapter 9 demonstrates the advantage of using a nuanced perspective on a domain's governing dynamics to develop a bespoke learning algorithm, rather than applying domain-agnostic tools in financial markets.

One may ask how "quantitative considerations of decision-making under uncertainty" [105] relate to trading in high-frequency financial markets? In particular, one may enquire as to how uncertainty and behavioural factors of participating agents may affect decision making? The paradigm promoted here is that of hierarchical causality [246]. While non-standard behaviour of agents in the financial system certainly contributes to the observed complexity, the focus of this thesis is derived from a *complexity economics* view, where agents with varying and adaptive objectives interact with the system and affect the states they perceive. This constitutes a broader perspective than an underlying assumption of rational agent behaviour, or any violations thereof, since *purpose* is the only requirement of agents under this paradigm [246]. A potential consequence of purposeful adaptive agents with different objectives may be the emergence of causal hierarchies that link various scales and structures of real markets via complex feedbacks. High-frequency trading is then one rather small component of such a hierarchy of causality, but one that has the benefit of large amounts of data that can be interrogated in order to probe the behaviour of a purposeful agent in such an adaptive system.

Appendix A

Derivation of the maximum likelihood function for explanatory power of cluster configuration

A.1 The Noh-Giada-Marsili coupling parameters

According to Noh [193], the generative model of the price associated with the ith stock can be written as

$$X_i(t) = g_{s_i}\eta_{s_i} + \sqrt{1 - g_{s_i}^2}\epsilon_i, \tag{A.1}$$

where the cluster-related influences are driven by η_{s_i} and the stock-specific influences by ϵ_i. Both innovations are treated as Gaussian random variables with unit variance and zero mean[1]. The relative contribution is controlled by the intra-cluster coupling parameter g_{s_i}. The Noh-Giada-Marsili model encodes the idea that stocks, say i and j, which have something in common belong in the same cluster, i.e. $s_i = s_j$. This comes

[1]This form of the price model ensures that the self correlation of a stock is one and independent of the cluster coupling. This can be seen by computing the self correlation $E[x_i^2]$ and using that clusters and stock unique process are unit variance zero mean processes

$$\mathrm{E}[(g_{s_i}\eta_{s_i} + \sqrt{1 - g_{s_i}^2}\epsilon_i)^2] = g_{s_i}^2 + (1 - g_{s_i}^2) = 1. \tag{A.2}$$

This is not a unique choice, another possible choice often used is

$$\mathrm{E}[(\frac{\sqrt{g_{s_i}}}{\sqrt{1 + g_{s_i}}}\eta_{s_i} + \frac{1}{\sqrt{1 + g_{s_i}}}\epsilon_i)^2] = \frac{1 + g_{s_i}}{1 + g_{s_i}} = 1. \tag{A.3}$$

with the caveat that stock membership in clusters is mutually exclusive and intra-cluster correlations are positive.

From Equation A.1 we compute the covariance for the ith and j^{th} stocks

$$\mathrm{E}[X_i(t)X_j(t)] = g_{s_i}^2 \mathrm{E}[\eta_{s_i}\eta_{s_j}] + (1 - g_{s_i}^2)\mathrm{E}[\epsilon_i\epsilon_j]. \tag{A.4}$$

Using the assumption of unit variance and zero mean for both the shared component (η_{s_i}) and stock component (ϵ_i) processes, the correlation between stock i and j is given by

$$C_{ij} = g_{s_i}^2 \delta_{s_i s_j} + (1 - g_{s_i}^2)\delta_{ij}. \tag{A.5}$$

The following cluster relations can be derived, where n_s is the number of stocks in the sth cluster and c_s is the internal correlation of the sth cluster, given that clusters are mutually exclusive

$$n_s = \sum_{i=1}^{N} \delta_{s_i s}, \qquad c_s = \sum_{i,j=1}^{N} C_{ij}\delta_{s_i s}\delta_{s_j s}. \tag{A.6}$$

From Equation A.5, for $s_i = s_j = s$, we have $C_{ij} \approx g_s^2$ ([103]). We can multiply both sides of Equation A.5 by $\delta_{s_i s}\delta_{s_j s}$ and sum over all i and j to find

$$\sum_{i,j} C_{ij}\delta_{s_i s}\delta_{s_j s} \quad = \quad \sum_{i,j} g_{s_i}^2 \delta_{s_i s_j}\delta_{s_i s}\delta_{s_j s} + \sum_{i,j}(1 - g_{s_i}^2)\delta_{ij}\delta_{s_i s}\delta_{s_j s}. \tag{A.7}$$

To sum out the delta functions over the clusters and stocks from

$$\sum_{i,j} C_{ij}\delta_{s_i s}\delta_{s_j s} \quad = \quad \sum_{i} \big(g_{s_i}^2 \delta_{s_i s} \sum_{j} \delta_{s_i s_j}\delta_{s_j s}\big) + \sum_{i} \big((1 - g_{s_i}^2)\delta_{s_i s} \sum_{j} \delta_{ij}\delta_{s_j s}\big),$$

we use that $\sum_j \delta_{ij}\delta_{s_i s} = \delta_{s_i s}$, $\sum_j \delta_{s_i s_j}\delta_{s_j s} = n_s\delta_{s_i s}$ and $\sum_i \delta_{s_i s}^2 = \sum_i \delta_{s_i s}$ to find

$$\sum_{i,j} C_{ij}\delta_{s_i s}\delta_{s_j s} = g_s^2 n_s \sum_i \delta_{s_i s} + (1 - g_s^2) \sum_i \delta_{s_i s}. \tag{A.8}$$

By combining Equations A.6 and A.8, we get

$$c_s = g_s^2 n_s^2 + (1 - g_s^2)n_s = g_s^2(n_s^2 - n_s) + n_s. \tag{A.9}$$

This is can be rearranged to finally obtain an expression for the intra-cluster coupling parameter for cluster s,

$$g_s = \sqrt{\frac{c_s - n_s}{n_s^2 - n_s}}. \tag{A.10}$$

A.2 The Noh-Giada-Marsili likelihood function

We evaluate the probability P of the data satisfying the model by using the multiplicative property of probabilities,

$$P(X_1(1),\ldots,X_N(D)) = \prod_{d=1}^{D}\prod_{i=1}^{N} P(X_i(d)). \tag{A.11}$$

The probability of being in a given state that satisfies the model is given as a delta function, such that we sum over all N stocks and all D features (date-times), taking expectations $\langle\ldots\rangle_{\eta,\epsilon}$ over the random processes associated with the stock-specific noise and the cluster-specific noise

$$P = \prod_{d=1}^{D}\left\langle \prod_{i=1}^{N} \delta\left(X_i(d) - (g_{s_i}\eta_{s_i} + \sqrt{1-g_{s_i}^2}\epsilon_i)\right)\right\rangle_{\eta,\epsilon}. \tag{A.12}$$

This takes on the form

$$\begin{aligned} P &= \prod_{d=1}^{D}\prod_{i=1}^{N}\int d\epsilon_i d\eta_{s_i} \exp\left[-\frac{1}{2}\sum_{k}^{N}\epsilon_k\delta_{ki}\epsilon_i - \frac{1}{2}\sum_{p,q}^{N}\eta_{s_p}\eta_{s_q}\delta_{s_p s_i}\delta_{s_q s_i}\right] && \text{(A.13)}\\ &\quad\times\delta\left(X_i(d) - g_{s_i}\eta_{s_i} - \sqrt{1-g_{s_i}^2}\epsilon_i\right). && \text{(A.14)}\end{aligned}$$

This is simplified to the following form, where the product over i stocks is converted to products of the clusters s and the n_s stocks in each cluster

$$\begin{aligned} P &= \prod_{s=1}^{S}\prod_{d=1}^{D}\int d\eta_s e^{-\frac{1}{2}\eta_s^2} && \text{(A.15)}\\ &\quad\times\prod_{i\in s}^{n_s}\int d\epsilon_i \exp\left[-\frac{1}{2}\epsilon_i^2\right]\delta\left(X_i(d) - g_s\eta_s - \sqrt{1-g_s^2}\epsilon_i\right). && \text{(A.16)}\end{aligned}$$

The Gaussian integral over the delta function is evaluated relative to the ϵ_i's, using that $\prod\int f(x)\delta(ax-x_0) = \prod\frac{1}{|a|}f(x_0/a)$ over the n_s delta functions,

$$P = \prod_{s=1}^{S}\prod_{d=1}^{D}\int \frac{d\eta_s}{(1-g_s^2)^{\frac{n_s}{2}}} e^{-\frac{1}{2}\eta_s^2}\prod_{i\in s}^{n_s}\exp\left[-\frac{1}{2}\frac{(g_s\eta_s - X_i)^2}{1-g_s^2}\right]. \tag{A.17}$$

Expanding out the integrand and using $\prod_i e_i^A = e^{\sum_i A_i}$,

$$P = \prod_{s=1}^{S}\prod_{d=1}^{D}\int \frac{d\eta_s}{(1-g_s^2)^{\frac{n_s}{2}}}\exp\left[-\frac{1}{2}\eta_s^2 - \frac{1}{2}\sum_{i\in s}^{n_s}\frac{(g_s^2\eta_s^2 - 2g_s\eta_s X_i + X_i^2)}{1-g_s^2}\right]. \tag{A.18}$$

Expanding out the sum terms and evaluating where possible

$$P = \prod_{s=1}^{S}\prod_{d=1}^{D}\int \frac{d\eta_s}{(1-g_s^2)^{\frac{n_s}{2}}} \exp\left[-\frac{1}{2}\eta_s^2 - \frac{1}{2}\frac{n_s g_s^2 \eta_s^2}{1-g_s^2}\frac{g_s\eta_s}{1-g_s^2}\sum_{i\in s}^{n_s} X_i - \frac{1}{2}\frac{1}{1-g_s^2}\sum_{i\in s}^{n_s} X_i^2\right].$$

This can be further simplified to

$$P = \prod_{s=1}^{S}\prod_{d=1}^{D}\int \frac{d\eta_s}{(1-g_s^2)^{\frac{n_s}{2}}} \exp\left[-\frac{1}{2}\frac{1-g_s^2+n_s g_s^2}{1-g_s^2}\eta_s^2 \frac{g_s\eta_s}{1-g_s^2}\sum_{i\in s}^{n_s} X_i - \frac{1}{2}\frac{1}{1-g_s^2}\sum_{i\in s}^{n_s} X_i^2\right].$$

We now evaluate the Gaussian integral using that $\int e^{-x^2}dx = \sqrt{\pi/2}$ and hence that $\int e^{-ax^2+bx}dx = \frac{\pi}{2a}e^{\frac{b^2}{4a}}$

$$\begin{aligned} P &= \prod_{s=1}^{S}\prod_{d=1}^{D} \frac{\sqrt{\pi}}{(1-g_s^2)^{\frac{n_s}{2}}}\frac{(1-g_s^2)^{\frac{1}{2}}}{(n_s g_s^2+(1-g_s^2))^{\frac{1}{2}}} \\ &\quad \times \exp\left[\frac{g_s^2}{2(n_s g_s^2+(1-g_s^2))(1-g_s^2)}(\sum_{i\in s}^{n_s} X_i)^2\right] \\ &\quad \times \exp\left[-\frac{1}{2}\frac{1}{1-g_s^2}\sum_{i\in s}^{n_s} X_i^2\right]. \end{aligned}$$

Evaluating the product of all D times, where $D >> 1$,

$$P = \prod_{s=1}^{S}\left[\frac{\sqrt{\pi}}{(1-g_s^2)^{\frac{n_s}{2}}}\frac{(1-g_s^2)^{\frac{1}{2}}}{(n_s g_s^2+(1-g_s^2))^{\frac{1}{2}}}\right]^D \tag{A.19}$$

$$\times \exp\left[\frac{g_s^2}{2(n_s g_s^2+(1-g_s^2))(1-g_s^2)}(\sum_{d}^{D}\sum_{i\in s}^{n_s} X_i)^2\right] \tag{A.20}$$

$$\times \exp\left[-\frac{1}{2}\frac{1}{1-g_s^2}\sum_{d}^{D}\sum_{i\in s}^{n_s} X_i^2\right]. \tag{A.21}$$

Using that $C_{ij} = \frac{1}{D}\sum_d X_i X_j$,

$$\sum_{d}^{D}(\sum_{i\in s} X_i)^2 = \sum_{i,j=1}^{N}(\sum_{d}^{D} X_i X_j)\delta_{s_i s}\delta_{s_j,s} = Dc_s, \tag{A.22}$$

and that the variance of the process in the sth cluster can be computed from the trace[2]

$$\sum_{i\in s}\sum_{d}^{D} X_i^2 = DC_{ii} = \sum_{i\in s}^{n_s} DC_{ii} = Dn_s. \tag{A.24}$$

[2]The trace of the correlation matrix for each cluster s can be verified from the eigenvalues

$$\sum_{i}^{N} C_{ii} = \sum_{s}\lambda_s = (n_s-1)(1-g_s^2) + n_s g_s^2 + (1-g_s^2) = n_s. \tag{A.23}$$

Substituting Equation A.22 and A.24 into Equation A.21,

$$P = \prod_{s=1}^{S}\left[\frac{\sqrt{\pi}}{(1-g_s^2)^{\frac{n_s}{2}}}\frac{(1-g_s^2)^{\frac{1}{2}}}{(n_s g_s^2+(1-g_s^2))^{\frac{1}{2}}}\right]^D \exp\left[-\frac{D}{2}\frac{n_s}{1-g_s^2}+\frac{D}{2}\frac{c_s}{1-g_s^2}\frac{g_s^2}{n_s g_s^2+(1-g_s^2)}\right]$$

We can rewrite this as

$$P = \prod_{s=1}^{S}\frac{\pi^{\frac{D}{2}}(n_s g_s^2+(1-g_s^2))^{\frac{-D}{2}}}{(1-g_s^2)^{\frac{D}{2}(n_s-1)}} \tag{A.25}$$

$$\times \exp\left[-\frac{D}{2}\frac{1}{1-g_s^2}\left(n_s-\frac{c_s g_s^2}{n_s g_s^2+(1-g_s^2)}\right)\right] \tag{A.26}$$

Then using that $P \propto e^{-DH_c}$, we can find $H_c \propto \ln(P)$ from Equation A.26, and using that $\ln\prod_i A_i = \sum_i \ln(A_i)$ to find the log-likelihood function [Need to use $D >> 1$ and look at expansion $(g_s - g_s^*)$]

$$\ln(P) = -\frac{D}{2}\sum_{s=1}^{S}\left[\ln(n_s g_s^2+(1-g_s^2)) \right. \tag{A.27}$$

$$+ \quad \left.(n_s-1)\ln(1-g_s^2)\right] \tag{A.28}$$

$$+ \quad \frac{D}{2}\sum_{s=1}^{S}[\ln(\pi)] \tag{A.29}$$

$$- \quad \frac{D}{2}\sum_{s=1}^{S}\frac{1}{1-g_s^2}\left[n_s-\frac{c_s g_s^2}{n_s g_s^2+(1-g_s^2)}\right]. \tag{A.30}$$

Using Equation A.10, we can substitute for g_s in A.10 to find the log-likelihood entirely in terms of n_s and c_s, using that $(1-g_s^2) = \frac{n_s^2-c_s}{n_s^2-n_s}$ and $\frac{c_s}{n_s} = n_s g_s^2 + (1-g_s^2)$:

$$H_c = \frac{1}{2}\sum_{s:n_s>0}\left[\log\frac{c_s}{n_s}+(n_s-1)\log\frac{n_s^2-c_s}{n_s^2-n_s}\right]+\frac{1}{2}\sum_{s:n_s>0}[\ln(\pi)+n_s]. \tag{A.31}$$

The last term is a constant, given that $\sum_{s:n_s>0} n_s = N$ where N is the number of objects. This is fixed for a given system. Hence the likelihood function required is

$$H_c = \frac{1}{2}\sum_{s:n_s>0}\left[\log\frac{c_s}{n_s}+(n_s-1)\log\frac{n_s^2-c_s}{n_s^2-n_s}\right] \tag{A.32}$$

up to a constant $\frac{1}{2}(S\ln(\pi)+N)$.

Bibliography

[1] Advanced Clustering Technologies: HPC cluster blog - GTX vs TESLA. http://www.headachefreehpc.com/company-blog/hpc-cluster-blog-gtx-vs-tesla.html. Accessed: 2014-09-25.

[2] Thomson Reuters Tick History API. https://customers.reuters.com/developer/Kits/TRTH/trth.aspx. Accessed: 2016-03-01.

[3] F. Abergel and A. Jedidi. Long time behaviour of a Hawkes process-based limit order book. *Working paper*, 2015. URL http://ssrn.com/abstract=2575498.

[4] F. Abergel, M. Anane, A. Chakraborti, A. Jedidi, and I.M. Toke. Limit order books. *Working paper*, 2015. URL http://fiquant.mas.ecp.fr/wp-content/uploads/2015/10/Limit-Order-Book-modelling.pdf.

[5] A. Adi, D. Botzer, G. Nechushtai, and G. Sharon. Complex event processing for financial services. *Proceedings from the IEEE Services Computing Workshops*, pages 7–12, 2006.

[6] A. Admati and P. Pfleiderer. A theory of intraday patterns: volume and price variability. *Review of Financial Studies*, **1**(1):3–40, 1988.

[7] A.R. Admati and P. Pfleiderer. Sunshine trading and financial market equilibrium. *Review of Financial Studies*, **4**(3):443–481, 1991.

[8] A. Alfonsi and A. Schied. Optimal trade execution and absence of price manipulations in limit order book models. *SIAM J. Financial Math.*, **1**(1):490–522, 2010.

[9] A. Alfonsi, A. Fruth, and A. Schied. Constrained portfolio liquidation in a limit order book model. *Banach Center Publ*, **83**:9–25, 2008.

[10] A. Alfonsi, A. Fruth, and A. Schied. Optimal execution strategies in limit order books with general shape functions. *Quantitative Finance*, **10**(2):143–157, 2010.

[11] A. Alfonsi, A. Schied, and A. Slynko. Order book resilience, price manipulation, and the positive portfolio problem. *SIAM J. Financial Math.*, **3**(1):511–533, 2012.

[12] R. Almgren. Optimal execution with non-linear impact functions and trading-enhanced risk. *Applied Mathematical Finance*, **10**:1–18, 2003.

[13] R. Almgren and N. Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, **3**:5–39, 2000.

[14] P.W. Anderson, K. Arrow, and D. Pines. *The economy as an evolving complex system (Santa-Fe Institute Series*. Westview Press, Boulder, Colorado, 1988.

[15] C. Arnsperger and Y. Varoufakis. What is neoclassical economics? the three axioms responsible for its theoretical oeuvre, practical irrelevance and, thus, discursive power. *Panoeconomicus*, **53**(1):5–18, 2006.

[16] W.B. Arthur. Complexity in economic and financial markets. *Complexity*, **1**(1): 20–25, 1995.

[17] W.B. Arthur. *Complexity and the economy*. Oxford University Press, Oxford, UK, 2014.

[18] W.B. Arthur, J.H. Holland, B. LeBaron, R. Palmer, and P. Taylor. Asset pricing under endogenous expectations in an artificial stock market. *The Economy as an Evolving Complex System*, **2**:15–44, 1997.

[19] E. Bacry and J.F. Muzy. Hawkes model for price and trades high-frequency dynamics. *Quantitative Finance*, **14**(7):1147–1166, 2014.

[20] E. Bacry and J.F. Muzy. Hawkes model for price and trades high-frequency dynamics. *Quantitative Finance*, **14**(7):1147–1166, 2014.

[21] E. Bacry and J.F. Muzy. Second order statistics characterization of Hawkes processes and non-parametric estimation. *Working paper*, 2015. URL `http://arxiv.org/pdf/1401.0903v2.pdf`.

[22] E. Bacry, K. Dayri, and J.F. Muzy. Non-parametric kernel estimation for symmetric Hawkes processes: Application to high frequency financial data. *The European Physical Journal B*, **85**(157), 2012.

[23] E. Bacry, T. Jaisson, and J.F. Muzy. Estimation of slowly decreasing Hawkes kernels: Application to high frequency order book modelling. *Working paper*, 2014. URL `http://arxiv.org/pdf/1412.7096.pdf`.

[24] E. Bacry, I. Mastromatteo, and J. Muzy. Hawkes processes in finance. *Market Microstructure and Liquidity*, 2015.

[25] J. Baker. Reducing bias and inefficiency in the selection algorithm. *Proceedings of the Second International Conference on Genetic Algorithms and their Application*, pages 14–21, 1987.

[26] F. Baldovin, F. Camana, M. Caporin, M. Caraglio, and A.L. Stella. Ensemble properties of high-frequency data and intraday trading rules. *Quantitative Finance*, **15**(2):231–245, 2015.

[27] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. *Proceedings from the International AAAI Conference on Weblogs and Social Media*, 2009.

[28] H. Bauke. Parameter estimation for power-law distributions by maximum likelihood methods. *The European Physical Journal B*, **58**(2):167–173, 2007.

[29] R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 1954.

[30] R. Bellman. A Markovian Decision Process. *Indiana University Mathematics Journal*, **6**(4):679–684, 1957.

[31] R. Bellman. *Dynamic programming*. Princeton University Press, Princeton, New Jersey, 1957.

[32] R. Bellman. *Adaptive control processes: A guided tour*. Princeton University Press, Princeton, New Jersey, 1961.

[33] R. Bellman and S. Dreyfus. *Applied dynamic programming*. Princeton University Press, Princeton, New Jersey, 1962.

[34] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Working paper*, 2014. URL `http://arxiv.org/pdf/1206.5538v3.pdf`.

[35] D. Bertsimas and A. Lo. Optimal control of execution costs. *Journal of Financial Markets*, **1**(1):1–50, 1998.

[36] B. Biais, P. Hillion, and C. Spatt. An empirical analysis of the limit order book and the order flow in the Paris Bourse. *Journal of Finance*, **50**:1655–1689, 1995.

[37] B. Biais, C. Glosten, and C. Spatt. Market microstructure: A survey of microfoundations, empirical results, and policy implications. *Journal of Financial Markets*, **8**(2):217–264, 2005.

[38] M. Blatt, S. Wiseman, and E. Domany. Superparamagnetic clustering of data. *Phys. Rev. Lett.*, **76**(18):3251–3254, 1996.

[39] M. Blatt, S. Wiseman, and E. Domany. Data clustering using a model granular magnet. *Neural Computation*, **9**:1805–1842, 1997.

[40] R. Bloomfield and M. O'Hara. Market transparency: Who wins and who loses? *Review of Financial Studies*, **12**(1):5–35, 1999.

[41] R. Bloomfield and M. O'Hara. Can transparent markets survive? *Journal of Financial Economics*, **55**(3):425–459, 2000.

[42] J.P. Bouchaud. Price impact. *Working paper*, 2009. URL `http://arxiv.org/pdf/0903.2428.pdf`.

[43] J.P. Bouchaud, Y. Gefen, M. Potters, and M. Wyart. Fluctuations and response in financial markets: The subtle nature of 'random' price changes. *Quantitative Finance*, **4**(2):176–190, 2004.

[44] J.P. Bouchaud, J.D. Farmer, and F. Lillo. *How markets slowly digest changes in supply and demand, in Handbook of Financial Markets: Dynamics and Evolution.* Elsevier, North-Holland, 2009.

[45] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *Proceedings from the International Conference on Machine Learning (ICML)*, 2012.

[46] C. Bowsher. Modelling security market events in continuous time: Intensity-based, multivariate point process models. *Journal of Econometrics*, **141**(2):876–912, 2005.

[47] W.A. Brock. Pathways to randomness in the economy: Emergent nonlinearity and chaos in economics and finance. *Estudios Economicos*, **8**:3–55, 1993.

[48] W.A. Brock and A. Kleidon. Periodic market closure and trading volume: A model of intraday bids and asks. *Journal of Economic Dynamics and Control*, **16**(3):451–489, 1992.

[49] A. Brodtkorb, T. Hagen, and M. Saetra. Graphics processing unit (GPU) programming strategies and trends in GPU computing. *Journal of Parallel and Distributed Computing*, **73**:4–13, 2012.

[50] A. Cartea, R.F. Donnelly, and S. Jaimungal. Enhancing trading strategies with order book signals. *Working paper*, 2015. URL `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2668277`.

[51] A. Cartea, S. Jaimungal, and J. Penalva. *Algorithmic and high-frequency trading.* Cambridge University Press, Cambridge, UK, 2015.

[52] L. Chan and J. Lakonishok. The behavior of stock prices around institutional trades. *Journal of Finance*, **50**(4):1147–1174, 1995.

[53] C. Chiarella. The cobweb model: Its instability and the onset of chaos. *Economic Modelling*, **5**(4):377–384, 1988.

[54] K. Chodorow. *MongoDB: The definitive guide.* O'Reilly, Sebastopol, California, 2013.

[55] D. Cieslakiewicz. Unsupervised asset cluster analysis implemented with parallel genetic algorithms on the Nvidia CUDA platform. Master's thesis, University of the Witwatersrand, 2014.

[56] D. Ciresan, U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[57] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *Working paper*, 2012. URL `http://arxiv.org/abs/1202.2745`.

[58] A. Clauset, C.R. Shalizi, and M.E.J. Newman. Power-law distributions in empirical data. *SIAM Review*, **51**(4):661–703, 2009.

[59] D. Coley. *An introduction to genetic algorithms for scientists and engineers.* World Scientific, 1999.

[60] D. Comer. The ubiquitous B-tree. *Computing Surveys*, **11**(2):121–137, 1979.

[61] R. Cont and P. Tankov. *Financial modelling with jump processes.* Chapman & Hall, CRC Financial Mathematics Series, 2004.

[62] M.M. Dacorogna, C.L. Gauvreau, U.A. Muller, R.B. Olsen, and O.V. Pictet. Changing time scale for short-term forecasting in financial markets. *Journal of Forecasting*, **15**:203–227, 1996.

[63] G. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, **20**(1):33–42, 2012.

[64] J. Danielsson and R. Payne. Measuring and explaining liquidity on an electronic limit order book - evidence from Reuters D2000-2. *Risk Measurement and Systematic Risk: Proceedings of the Third Joint Central Bank Research Conference, Bank for International Settlements, Basel*, 2002.

[65] F. Darema. SPMD model: Past, present and future. *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 8th European PVM/MPI Users' Group Meeting*, 2001.

[66] A. Dassios and H. Zhao. Exact simulation of Hawkes process with exponentially decaying intensity. *Electronic Communications in Probability*, **18**(62):1–13, 2013.

[67] P. Dayan and C. Watkins. *Reinforcement learning.* Encyclopedia of Cognitive Science, 2001.

[68] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, **51**(1):107–113, 2008.

[69] H. Degryse, F. deJong, M. Ravenswaaij, and G. Wuyts. Aggressive orders and the resiliency of a limit order market. *Review of Finance*, **9**(2):201–242, 2005.

[70] H. Demsetz. The cost of transacting. *Quarterly Journal of Economics*, **82**:33–53, 1968.

[71] E. Derman. The perception of time, risk and return during periods of speculation. *Quantitative Finance*, **2**:282–296, 2002.

[72] T. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Abstraction, Reformulation and Approximation*, pages 26–44, 2000.

[73] J. Digalakis and K. Margaritis. Parallel evolutionary algorithms on Message-Parsing clusters. *Proceedings from the Parallel Computing Conference*, 2003.

[74] B. Du Preez. JSE market microstructure. Master's thesis, University of the Witwatersrand, 2013.

[75] D. Easley and M. O'Hara. Price, trade size and information in securities markets. *Journal of Financial Economics*, **19**:69–90, 1987.

[76] D. Easley, M.M. López de Prado, and M. O'Hara. The volume clock: Insights into the high-frequency paradigm (Digest summary). *Journal of Portfolio Management*, **39**(1):19–29, 2012.

[77] S.F. Edwards and P.W. Anderson. Theory of spin glasses. *J. Phys. F: Met. Phys*, **5**:965–974, 1975.

[78] F. Emmert-Streib and M. Dehmer. Influence of the time scale on the construction of financial networks. *PLoS ONE*, **5**(9), 2010.

[79] R.F. Engle and J.R. Russell. Autoregressive conditional duration: A new model for irregularly spaced transaction data. *Econometrica*, **66**:1127–1162, 1998.

[80] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, **6**:503–556, 2005.

[81] M. Fard and J. Pineau. MDPs with non-deterministic policies. *Advances in Neural Information Processing Systems*, **21**:1065–1073, 2009.

[82] J.D. Farmer and F. Lillo. On the origin of power-law tails in price fluctuations. *Quantitative Finance*, **4**(1):7–11, 2004.

[83] V. Filimonov and D. Sornette. Quantifying reflexivity in financial markets: Toward a prediction of flash crashes. *Physical Review E*, **85**(5):1065–1073, 2012.

[84] V. Filimonov and D. Sornette. Apparent criticality and calibration issues in the Hawkes self-excited point process model: Application to high-frequency financial data. *Working paper*, 2013. URL `http://arxiv.org/abs/1308.6756`.

[85] D.K. Foley. A statistical equilibrium theory of markets. *Journal of Economic Theory*, **62**(2):321–345, 1994.

[86] H. Föllmer. Random economies with many interacting agents. *Journal of Mathematical Economics*, **1**(1):51–62, 1974.

[87] H. Föllmer. Spatial risk measures and their local specification: The locally law-invariant case. *Statistics & Risk Modelling*, **31**(1):79–101, 2014.

[88] M. Forster and T. George. Anonymity in securities markets. *Journal of Financial Intermediation*, **1**.

[89] P.A. Forsyth. A Hamilton-Jacobi-Bellman approach to optimal trade execution. *Applied Numerical Mathematics*, **61**(2):241–265, 2011.

[90] B. Franke, J.F. Plante, R. Roscher, A. Lee, C. Smyth, A. Hatefi, F. Chen, E. Gil, A. Schwing, A. Selvitella, M.M. Hoffman, R. Grosse, D. Hendricks, and N. Reid. Statistical inference, learning and models in big data. *International Statistical Review (accepted, to appear)*, 2015. URL `http://arxiv.org/abs/1509.02900`.

[91] C. Frei and N. Westray. Optimal execution of a VWAP order: A stochastic control approach. *Mathematical Finance*, **25**(3):612–639, 2015.

[92] K. French and R. Roll. Stock return variances: The arrival of information and the reaction of traders. *Journal of Financial Economics*, **17**(1):5–26, 1986.

[93] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software - practice and experience*, **21**(11):1129–1164, 1991.

[94] X. Gabaix, P. Gopikrishnan, V. Plerou, and H.E. Stanley. A theory of power-law distributions in financial market fluctuations. *Nature*, **423**(6937):267–270, 2003.

[95] T. Galla and J.D. Farmer. Complex dynamics in learning complicated games. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, **110**(4):1232–1236, 2013.

[96] F. Garcia and S. Ndiaye. A learning rate analysis of reinforcement learning algorithms in finite-horizon. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

[97] M.B. Garman. Market microstructure. *Journal of Financial Economics*, **3**:257–275, 1976.

[98] J. Gatheral. No-dynamic-arbitrage and market impact. *Quantitative Finance*, **10** (7):749–759, 2010.

[99] J. Gatheral and A. Schied. Optimal trade execution under geometric Brownian motion in the Almgren and Chriss framework. *International Journal of Theoretical and Applied Finance*, **14**(3):353–368, 2011.

[100] J. Gatheral and A. Schied. Dynamical models of market impact and algorithms for order execution. *Handbook on Systemic Risk*, pages 579–599, 2013. URL `http://dx.doi.org/10.2139/ssrn.2034178`.

[101] T. Gebbie, D. Wilcox, and B. Mbambiso. Spin, stochastic factor models, and a GA. *Southern African Finance Association Conference*, 2010.

[102] R. Gençay, N. Gradojevic, F. Selçuk, and B. Whitcher. Asymmetry of information flow between volatilities across time scales. *Quantitative Finance*, **10**(8):895–915, 2010.

[103] L. Giada and M. Marsili. Data clustering and noise undressing of correlation matrices. *Phys. Rev. E*, **63**(1), 2001.

[104] L. Giada and M. Marsili. Algorithms of maximum likelihood data clustering with applications. *Physica A*, **315**(34):650–664, 2002.

[105] I. Gilboa. *Theory of decision under uncertainty (Econometric Society Monographs)*. Cambridge University Press, Cambridge, UK, 2009.

[106] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. *Proceedings from the International Conference on Machine Learning (ICML)*, 2011.

[107] L.R. Glosten and P. Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed agents. *Journal of Financial Economics*, **14**:71–100, 1985.

[108] M.K. Goldberg, M. Hayvanovych, and M. Magdon-Ismail. Measuring similarity between sets of overlapping clusters. *Proceedings from the International Conference on Social Computing (SocialCom)*, pages 303–308, 2010.

[109] M.D. Gould, M.A. Porter, S. Williams, M. McDonald, D.J. Fenn, and S.D. Howison. Limit order books. *Working paper*, 2013. URL `http://arxiv.org/abs/1012.0349`.

[110] J.E. Griffin and R.C.A. Oomen. Covariance measurement in the presence of non-synchronous trading and market microstructure noise. *Journal of Econometrics*, **160**:58–68, 2011.

[111] J.E. Griffin, F. Nardari, and R. Stulz. Are daily cross-border equity flows pushed or pulled? *The Review of Economics and Statistics*, **86**(3):641–657, 2004.

[112] S. Hardiman and J.P. Bouchaud. Branching ratio approximation for the self-exciting Hawkes process. *Working paper*, 2014. URL `http://arxiv.org/abs/1403.5227`.

[113] S. Hardiman, N. Bercot, and J.P. Bouchaud. Critical reflexivity in financial markets: A Hawkes process analysis. *The European Physica Journal B*, **86**(10):1–9, 2013.

[114] L. Harris. An transaction data study of weekly and intradaily patterns in stock returns. *Journal of Financial Economics*, **16**(1):99–117, 1986.

[115] M. Harvey, D. Hendricks, T. Gebbie, and D. Wilcox. Deviations in expected price impact for small transaction volumes under fee restructuring. *Working paper*, 2016. URL `http://arxiv.org/abs/1602.04950`.

[116] J. Hasbrouck. Trades, quotes, inventories and information. *Journal of Financial Economics*, **22**:229–252, 1988.

[117] J. Hasbrouck. Measuring the information content of stock trades. *Journal of Finance*, **46**:179–207, 1991.

[118] J. Hasbrouck. Security bid-ask dynamics with discreteness and clustering: Simple strategies for modeling and estimation. *Journal of Financial Markets*, **2**:1–28, 1999.

[119] J. Hasbrouck. High-frequency quoting: Short-term volatility in bids and offers. *Working paper*, 2015. URL `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2237499`.

[120] A.G. Hawkes. Spectra of some self-exciting and mutually-exciting point processes. *Biometrika*, **58**(1):83–90, 1971.

[121] D. Hendricks. An online learning algorithm with scale-specific state space enumeration for optimal trade execution in high-frequency markets. *Working paper*, 2016.

[122] D. Hendricks. Using real-time cluster configurations of streaming asynchronous features as online state descriptors in financial markets. *Working paper*, 2016. URL `http://arxiv.org/abs/1603.06805`.

[123] D. Hendricks and M. Harvey. Reconciling order book resiliency and price impact. *Working paper*, 2016.

[124] D. Hendricks and D. Wilcox. A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution. *Proceedings from IEEE Conference on Computational Intelligence for Financial Economics and Engineering*, 2014. URL `http://dx.doi.org/10.1109/CIFEr.2014.6924109`.

[125] D. Hendricks, T. Gebbie, and D. Wilcox. High-speed detection of emergent market clustering via an unsupervised parallel genetic algorithm. *South African Journal of Science*, 112(1/2), 2016. URL `http://dx.doi.org/10.17159/sajs.2016/20140340`.

[126] D. Hendricks, T. Gebbie, and D. Wilcox. Detecting intraday financial market states using temporal clustering. *Quantitative Finance*, 2016. URL `http://dx.doi.org/10.1080/14697688.2016.1171378`.

[127] G. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, **11**(10):428–434, 2007.

[128] G. Hinton, L. Deng, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, **29**(6):82–97, 2012.

[129] D.C. Hoaglin, F. Mosteller, and J.W. Tukey. *Understanding robust and exploratory data analysis*. Wiley series in probability and mathematical statistics: Applied probability and statistics, 1983.

[130] J.H. Holland. Complex adaptive systems. *Daedalus*, **121**(1):17–30, 1992.

[131] R. Holthausen, R. Leftwich, and D. Mayers. Large-block transactions, the speed of response and temporary and permanent stock-price effects. *Journal of Financial Economics*, **26**(1):71–95, 1990.

[132] C.H. Hommes. Financial markets as nonlinear adaptive evolutionary systems. *Quantitative Finance*, **1**(1):149–167, 2001.

[133] R. Howard. *Dynamic programming and Markov processes*. MIT Press, Cambridge, Massachusetts, 1960.

[134] G. Huberman and W. Stanzl. Optimal liquidity trading. *Yale School of Management Working Paper*, 2001.

[135] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, **14**(1):55–63, 1968.

[136] G. Iori, M.G. Daniels, J.D. Farmer, L. Gillemot, S. Krishnamurthy, and E. Smith. An analysis of price impact function in order-driven markets. *Physica A: Statistical Mechanics and its Applications*, **324**:146–151, 2003.

[137] E. Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, **31**(1):253–258, 1925.

[138] M. Ismail. Parallel Genetic Algorithms (PGAs): Master-Slave paradigm approach using MPI. *IEEE e-Tech*, **31**:83–87, 2004.

[139] T. Jaakkola, M.I. Jordan, and S.P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, **6**(6):1185–1201, 1994.

[140] S. Jaimungal, A. Cartea, and J. Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, Cambridge, UK, 2015.

[141] P.C. Jain and G.H. Joh. The dependence between hourly prices and trading volume. *The Journal of Financial and Quantitative Analysis*, **23**(3):269–283, 1988.

[142] T. Jaisson and M. Rosenbaum. Limit theorems for nearly unstable Hawkes processes. *The Annals of Applied Probability*, **25**(2):600–631, 2015.

[143] W.S. Jevons. *The theory of political economy*. Macmillan and Co., London and New York, 1871.

[144] B. Johnson. *Algorithmic trading and DMA: An introduction to direct access trading strategies*. 4Myeloma Press, London, 2010.

[145] JSE. Dual-listed companies (retrieved: 08/03/2014). 2014. URL `http://www.jse.co.za/how-tolist/main-board/dual-listed-companies.aspx`.

[146] JSE. Market data - Equities, derivatives and interest rate products price list (retrieved: 13/07/2015). 2015. URL `http://www.jse.co.za/services/market-data`.

[147] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, **4**:237–285, 1996.

[148] N. Kaldor. A classificatory note on the determinateness of equilibrium. *Review of Economic Studies*, **1**(2):122–136, 1934.

[149] M. Karacuka and A. Zaman. The empirical evidence against neoclassical utility theory: A review of the literature. *Int. J. of Pluralism and Economics Education*, **3**(4):366–414, 2012.

[150] M. Kirchner. An estimation procedure for the Hawkes process. *Working paper*, 2015. URL `http://arxiv.org/pdf/1509.02017v1.pdf`.

[151] A. Kirman. Ants, rationality and recruitment. *The Quarterly Journal of Economics*, **108**(1):137–156, 1993.

[152] A. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *G. Ist. Ital. Attuari*, **4**:83–91, 1933.

[153] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. *Proceedings from Neural Information Processing Systems (NIPS) conference*, 2012.

[154] L. Kullmann, J. Kertész, and R. Mantegnae. Identification of clusters of companies in stock indices via Potts super-paramagnetic transitions. *Working paper*, 2000. URL `http://arxiv.org/abs/cond-mat/0002238`.

[155] D. Kwiatowski, P.C.B. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54(1-3):159–178, 1992.

[156] A.S. Kyle. Continuous auctions and insider trading. *Econometrica*, **53**(6):1315–1336, 1985.

[157] M. Lallouache and D. Challet. The limits of statistical significance of Hawkes processes fitted to financial data. *Quantitative Finance*, **16**(1):1–11, 2016.

[158] S. Lange, T. Gabel, and M. Riedmiller. *"Batch reinforcement learning" in reinforcement learning*. Springer, Berlin Heidelberg, 2012.

[159] J. Large. Measuring the resiliency of an electronic limit order book. *Journal of Financial Markets*, **10**:1–25, 2007.

[160] S. Laruelle, C.A. Lehalle, and G. Pagés. Optimal split of orders across liquidity pools: A stochastic algorithm approach. *SIAM Journal of Financial Mathematics*, **2**:1042–1076, 2011.

[161] S. Laruelle, C.A. Lehalle, and G. Pagés. Optimal posting price of limit orders: Learning by trading. *Mathematics and Financial Economics*, **7**(3):359–403, 2013.

[162] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings from the International Conference on Machine Learning (ICML)*, 2009.

[163] P.A. Lewis and G.S. Shedler. Simulation of nonhomogeneous Poisson processes by thinning. *Naval Research Logistics Quarterly*, **26**(3):403–413, 1979.

[164] F. Lillo, J.D. Farmer, and R.N. Mantegna. Econophysics: Master curve for price-impact function. *Nature*, **421**:129–130, 2003.

[165] G.M. Ljung and G.E.P. Box. On a measure of a lack of fit in time series models. *Biometrika*, **65**(2):297–303, 1978.

[166] C. Lorenz and A. Schied. Drift dependence of optimal trade execution strategies under transient price impact. *Finance and Stochastics*, **17**(4):743–770, 2013.

[167] A. Madhavan. Security prices and market transparency. *Journal of Financial Intermediation*, **5**(3):255–283, 1996.

[168] A. Madhavan. Market microstructure: A survey. *Journal of Financial Markets*, **3** (3):205–258, 2000.

[169] C. Malherbe. Fourier method for the measurement of univariate and multivariate volatility in the presence of high frequency data. Master's thesis, University of Cape Town, 2007. URL `http://www.mth.uct.ac.za/academics/postgrad/graduatethesis/MSc_Chanel_Malherbe.pdf`.

[170] P. Malliavin and M.E. Mancino. Fourier series method for measurement of multivariate volatilities. *Finance and Stochastics*, **6**:49–61, 2002.

[171] P. Malliavin and M.E. Mancino. A Fourier transform method for nonparametric estimation of multivariate volatility. *Annals of Statistics*, **37**(4):1983–2010, 2009.

[172] M. Marsili. Dissecting financial markets: Sectors and states. *Quantitative Finance*, **2**(4):297–302, 2002.

[173] R. Martins. The statistical significance of mutually-exciting Hawkes processes fitted to JSE data. AMF Honours Project, University of the Witwatersrand, 2015. *Supervised by D. Hendricks.*

[174] R. Martins and D. Hendricks. The statistical significance of mutltivariate hawkes processes fitted to limit order book data. *Working paper*, 2016. URL `http://arxiv.org/abs/1604.01824`.

[175] I. Mastromatteo and M. Marsili. On the criticality of inferred models. *Journal of Statistical Mechanics: Theory and Experiment*, **2011**(10), 2011.

[176] K. Mazibuko. Quantifying resiliency of the JSE limit order book following large trades. AMF Honours Project, University of the Witwatersrand, 2014. *Supervised by D. Hendricks.*

[177] B. Mbambiso. Dissecting the South African equity markets into sectors and states. Master's thesis, University of Cape Town, 2009.

[178] T.H. McInish and R.A. Wood. An analysis of intraday patterns in bid/ask spreads for NYSE stocks. *The Journal of Finance*, **47**(2):753–764, 1992.

[179] G.J. McLachlan, D. Peel, and W.J. Whiten. Maximum likelihood clustering via normal mixture models. *Signal Processing: Image Communication*, **8**(2):105–111, 1996.

[180] A. McNeil, R. Frey, and P. Embrechts. *Quantitative Risk Management: Concepts, techniques and tools.* Princeton University Press, Princeton Series in Finance, 2015.

[181] J. McPartland. Recommendations for equitable allocation of trades in high frequency trading environments. *Federal Reserve Bank of Chicago Policy Discussion Paper*, 2013. URL `https://www.chicagofed.org/~/media/publications/policy-discussion-papers/2013/pdp2013-01-original-pdf.pdf`.

[182] F. Melo. Convergence of Q-learning: A simple proof. *Institute of Systems and Robotics, Technical Report*, 2001.

[183] C. Menger. *Grundsätze der Volkswirtschaftslehre.* Braumüller, 1871.

[184] T. Minney. African Capital Market News: London and Johannesburg stock exchanges migrate to Millennium Exchange system (retrieved: 01/02/2016). 2011. URL `http://www.africancapitalmarketsnews.com/906/london-and-johannesburg-stock-exchanges-migrate-to-millennium-exchange-system/`.

[185] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *Neural Information Processing (NIPS) deep learning workshop*, 2013.

[186] E. Moro, J. Vicente, L.G. Moyano, A. Gerig, J.D. Farmer, G. Vaglica, F. Lillo, and R.N. Mantegna. Market impact and trading profile of hidden orders in stock markets. *Phys. Rev. E*, **80**:066102, 2009.

[187] U.A. Müller, M.M. Dacorogna, R.D. Davé, R.B. Pictet, O.V. Olsen, and J.R. Ward. Fractals and intrinsic time — a challenge to econometricians. *Olsen and Associates, Zurich*, 1995.

[188] M. Mungan and J.J. Ramasco. Stability of maximum-likelihood-based clustering methods: Exploring the backbone of classifications. *Journal of Statistical Mechanics: Theory and Experiment*, **4**, 2010.

[189] L. Nan, G. Pengdong, L. Yongquan, and Y. Wenhua. The implementation and comparison of two kinds of parallel genetic algorithm using Matlab. *Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, 2010.

[190] E. Neuman and A. Schied. Optimal portfolio liquidation in target zone models and catalytic superprocesses. *Finance and Stochastics*, **20**(2):495–509, 2016.

[191] Y. Nevmyvaka. *Normative approach to market microstructure analysis*. PhD thesis, Carnegie Mellon University, 2004.

[192] Y. Nevmyvaka, Y. Feng, and M. Kearns. Reinforcement learning for optimal trade execution. *Proceedings of the 23rd international conference on machine learning*, 2006.

[193] J. Noh. A model for correlations in stock markets. *Physical Review E*, **61**, 2000.

[194] JSE Market Notices and Circulars. New equity market trading and information solution and new SENS system - Go live readiness confirmed for Monday 2 July 2012 (retrieved: 01/02/2016). 2012. URL `https://www.jse.co.za/content/JSENoticesandCircularsItems/Equity%20Markets/2012/20120629-062.pdf`.

[195] JSE Market Notices and Circulars. JSE equity market transaction billing model methodology change notice, JSE market notice no. 136 (retrieved: 01/02/2016). 2013. URL https://www.jse.co.za/content/JSENoticesandCircularsItems/Equity%20Markets/2013/2013_136.pdf.

[196] JSE Market Notices and Circulars. Equity market price list (retrieved: 01/02/2016). 2013. URL https://www.jse.co.za/content/JSENoticesandCircularsItems/Equity%20Markets/2012/20121130-098B-.pdf.

[197] JSE Market Notices and Circulars. JSE colocation services go live 12 May 2014 (retrieved: 01/02/2016). 2014. URL https://www.jse.co.za/content/JSEHotlinesItems/JSE%20Service%20Hotline%209014%20JSE%20Colocation%20Service%20Go%20live%2012%20May%202014.pdf.

[198] JSE Market Notices and Circulars. The lowest-latency connection to JSE markets: Colocation (retrieved: 01/02/2016). 2014. URL https://www.jse.co.za/content/JSETechnologyDocumentItems/3.%20JSE%20Colocation%20Brochure%202015.pdf.

[199] JSE Market Notices and Circulars. Equity market price list 2014 v1.1 (retrieved: 01/02/2016). 2014. URL https://www.jse.co.za/content/JSENoticesandCircularsItems/Equity%20Markets/2014/218B.pdf.

[200] Nvidia. *Nvidia CUDA C Programming Guide*. Nvidia Corporation, 2011.

[201] Nvidia. *SLI Best Practices*. Nvidia Corporation, 2011. URL http://developer.download.nvidia.com/whitepapers/2011/SLI_Best_Practices_2011_Feb.pdf.

[202] Nvidia. *CUDA Dynamic Parallelism Programming Guide*. Nvidia Corporation, 2012.

[203] Nvidia. *CUDA C Best Practices Guide*. Nvidia Corporation, 2012.

[204] Y. Ogata. Statistical models for earthquake occurences and residual analysis for point processes. *Journal of the American Statistical Association*, **83**(401):9–27, 1988.

[205] Y. Ogata. Seismicity analysis through point process modelling: A review. *Pure and Applied Geophysics*, **155**(2):471–507, 1999.

[206] M. O'Hara. *Market Microstructure Theory*. Blackwell publishing, 1998.

[207] T. Ozaki. Maximum likelihood estimation of Hawkes self-exciting point process. *Annals of the Institute of Statistical Mathematics*, **31**(1):145–155, 1979.

[208] D.A. Patterson and J.L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface, Fifth edition*. Morgan Kaufmann, 2013.

[209] A. Perold. The implementation shortfall: Paper vs reality. *Journal of Portfolio Management*, **14**(3):4–9, 1988.

[210] V. Plerou, H.E. Stanley, X. Gabaix, and P. Gopikrishnan. On the origin of power-law fluctuations in stock prices. *Quantitative Finance*, **4**(1):11–15, 2004.

[211] P. Pospichal, J. Jaros, and J. Schwarz. Parallel genetic algorithm on the CUDA architecture. *Proceedings of the 2010 International Conference on Applications of Evolutionary Computation*, pages 442–451, 2010.

[212] M. Potters and J.P. Bouchaud. More statistical properties of order books and price impact. *Physica A: Statistical Mechanics and its Applications*, **324**(1-2):133–140, 2003.

[213] M. Puterman. *Markov Decision Processes*. John Wiley and Sons, 1994.

[214] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 2005.

[215] M. Rambaldi, E. Bacry, and F. Lillo. The role of volume in order book dynamics: A multivariate Hawkes process analysis. *Working paper*, 2016. URL `http://arxiv.org/pdf/1602.07663.pdf`.

[216] S. Ross. *Introduction to stochastic dynamic programming*. Academic Press, New York, 1983.

[217] J.B. Rosser Jr. *Complexity in economics*. Edward Elgar Publishing, Cheltenham, England, 2004.

[218] A. Schied. Robust strategies for optimal order execution in the Almgren–Chriss framework. *Applied Mathematical Finance*, **20**(3):264–286, 2013.

[219] A. Schied and T. Schöneborn. Risk aversion and the dynamics of optimal liquidation strategies in illiquid markets. *Finance and Stochastics*, **13**(2):181–204, 2009.

[220] A. Schied and T. Zhang. A state-constrained differential game arising in optimal portfolio liquidation. *Mathematical Finance*, 2015. URL `http://dx.doi.org/10.1111/mafi.12108`.

[221] A. Schied, T. Schöneborn, and M. Tehranchi. Optimal basket liquidation for CARA investors is deterministic. *Applied Mathematical Finance*, **17**(6):471–489, 2010.

[222] T. Schöneborn and A. Schied. Liquidation in the face of adversity: Stealth vs. sunshine trading. *EFA 2008 Athens Meetings Paper*, 2009.

[223] S. Sivanandam and S. Deepa. *Introduction to Genetic Algorithms*. Springer, 2010.

[224] S. Smidt. Which road to an efficient stock market: Free competition or regulated monopoly? *Financial Analysts Journal*, **27**(5):64–69, 1971.

[225] N. Smirnov. Table for estimating the goodness of fit of empirical distributions. *Annals of Mathematical Statistics*, **19**:279–281, 1948.

[226] D.L. Stein. Spin glasses: Still complex after all these years? *Santa Fe Institute Working Paper*, 2003.

[227] D.L. Stein and C.M. Newman. Spin glasses: Old and new complexity. *Complex Systems*, **20**(2):115–126, 2011.

[228] D. Sussman. The replica approach: Spin and structural glasses. *Working paper*, 2008. URL `http://guava.physics.uiuc.edu/~nigel/courses/569/Essays_Fall2008/files/sussman.pdf`.

[229] R. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224, 1990.

[230] R. Sutton and A. Barto. *Reinforcement learning*. MIT Press, Cambridge, MA, 1998.

[231] E.L. Thorndike. *Animal intelligence: Experimental studies*. The Macmillan Company, New York, USA, 1911.

[232] I.M. Toke and F. Pomponio. Modelling trades-through in a limit order book using Hawkes processes. *Economics: The Open-Access, Open-Assessment E-Journal*, **6** (22):1–23, 2012.

[233] J.W. Tukey. The future of data analysis. *Annals of Mathematical Statistics*, **33** (1):1–67, 1961.

[234] J.W. Tukey. *Exploratory Data Analysis*. Behavioural Science, Pearson, 1st edition, 1977.

[235] J.W. Tukey. Exploratory Data Analysis: Past, present and future. *Technical Report No. 302 (Series 2)*, 1993. URL `http://www.dtic.mil/dtic/tr/fulltext/u2/a266775.pdf`.

[236] D. Vayanos. Strategic trading and welfare in a dynamic market. *Review of economic studies*, **66**:219–254, 1999.

[237] D. Vayanos. Strategic trading in a dynamic noisy market. *Journal of Finance*, **56** (1):131–171, 2001.

[238] T. Veblen. Why is economics not an evolutionary science? *Quarterly Journal of Economics*, **12**(4):373–397, 1898.

[239] T. Veblen and J. Boulton. Why is economics not an evolutionary science? *Emergence: Complexity and Organization*, **12**(2):41–69, 2010.

[240] D. Vere-Jones. Stochastic models for earthquake occurence. *Journal of the Royal Statistical Society, Series B (Methodological)*, **32**(1):1–62, 1970.

[241] L. Walras. *Éléments d'économie politique pure; ou, Théorie de la richesse sociale.* F. Rouge, 1896.

[242] S. Wang and R.H. Swendsen. Cluster Monte Carlo algorithms. *Physica A*, **167** (565), 1990.

[243] C. Watkins. *Learning from delayed rewards.* PhD thesis, Cambridge University, 1989.

[244] C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, **8**: 279–292, 1992.

[245] C. White and D. White. Markov Decision Processes. *European Journal of Operational Research*, **39**:1–16, 1989.

[246] D. Wilcox and T. Gebbie. Hierarchical causality in financial economics. *Working paper*, 2014. URL `http://ssrn.com/abstract=2544327`.

[247] D. Wilcox, D. Hendricks, and T. Gebbie. Fourier methods for correlation estimation. *Working paper*, 2016.

[248] M. Wilinski, W. Cui, and A. Brabazon. An analysis of price impact functions of individual trades on the London Stock Exchange. *Proceedings from IEEE Conference on Computational Intelligence for Financial Economics and Engineering*, 2014. URL `http://dx.doi.org/10.1109/CIFEr.2014.6924047`.

[249] S. Wiseman, M. Blatt, and E. Domany. Superparamagnetic clustering of data. *Phys. Rev. E*, **57**:37–67, 1998.

[250] R.A. Wood, T.H. McInish, and J.K. Ord. An investigation of transactions data for NYSE stocks. *The Journal of Finance*, **40**(3):723–739, 1985.

[251] F.Y. Wu. The Potts model. *Reviews of Modern Physics*, **54**(1):235–268, 1982.

[252] S. Xiao and W. Feng. Inter-Block GPU communication via Fast Barrier Synchronization. *2010 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–12, 2010.

[253] A.G. Zawadowski, J. Kertèsz, and G. Andor. Large price changes on small scales. *Physica A: Statistical Mechanics and its Applications*, **344**(1-2):221–226, 2004.

[254] G. Zhang, W. Liu, and G. Liu. Parallel genetic algorithm based on the MPI environment. *Telkomnika*, **10**:1708–1715, 2012.

[255] L. Zhang, P.A. Mykland, and Y. Aït-Sahalia. A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *Journal of the American Statistical Association*, **100**(472):1394–1411, 2005.

[256] S. Zhang and Z. He. Implementation of parallel genetic algorithm based on CUDA. *Proceedings of the 4th International Symposium on Advances in Computation and Intelligence*, pages 24–30, 2009.