Calculating the price response of stocks in emerging markets

E. T. Nonyane: 599795
Supervisor: Prof. T. Gebbie

Programme in Computational and Applied Mathematics,
School of Computer Science and Applied Mathematics,

UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

_____

Signature

_____

Date

Acknowledgements

I would like to start by thanking Prof. T. Gebbie for affording me the opportunity to do this research project under his supervision. I would like to thank him for setting time aside for weekly consultations and continued interactions. I would also like to thank him for arranging weekly market microstructure seminars in the semester half of 2017.

Secondly, I would like to thank Prof. D. Wilcox and Prof T. Gebbie for providing me with resources to aid in the completion of my project. I would like to thank them for allowing me to use STELLA (computing solution), BELLA (database server) and SUPERALLA (computing solution). Furthermore, I would to thank them for allowing me to use the QuERILAb office.

From the QuERILab group, I would like to thank Prof T. Gebbie, Prof. D. Wilcox, Andrew Paskaramoorthy, Donovan Platt, Nicolas Murphy, Lionel Yelibi, Charlene Chipoyera and Washington Mhlanga. I would like to thank this group for attending some the QuERILAb market microstructure seminar and sharing the details of their research areas. I would also like to thank them for the discussions that we had related to certain aspects my project.

Lastly, I would like to thank Prof Diane Wilcox, Dr Dieter Hendricks, Mr Michael Harvey and, Mr Brett Du Preez for paving the way for this project. Without the groundwork laid by these authors, this project would not have gone as seamlessly as it did.

Abstract

In stock markets, *price response* refers the change in the market price subsequent to a trade. It can result in unintended costs and lead to a substantial amount of risk. To quantify the effect of price response, several measures of it—such as *price impact* and *bare response*—have introduced and analysed in the literature. This dissertation uses tick data from the Thompson Reuters Tick History database and some of the previously introduced price response measure to examine the price response of trades for a selection of emerging market stocks. Although price response is widely discussed and measured for the European and North American stock markets, few studies of it exit for emerging markets. Thus, we attempt to fill the gap in the literature by considering stocks from Brazil, Russia, India, China, South Africa and Kenya and Egypt. We suggest and document a highly scalable and reproducible work-flow for calculating price response for these markets. In accord with the findings in developed markets, we find that the price response of trades tends to increase with traded volume and decrease over time. We also find that the most actively traded stocks (those with a high average daily volume or average daily traded value) tend to exhibit a lower degree of price response.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Open outcry was the basis for trading securities in early stock exchanges. Prices were negotiated and recorded manually and stockbrokers used hand signals and shouts to communicate information about orders [1]. The trading mechanism was relatively simple: brokers on trading floors acted as a conduit between buyers and sellers and transactions were settled face to face [2]. Stocks were issued on paper and market data was disseminated through news papers, tickers tapes and even carrier pigeons at some point in time [3, 4]. In all likelihood, managing and analysing market data was arduous. In 1919, the New York Stock Exchange (NYSE) closed for three days to allow back office workers to catch up on work, and between 1967 and 1970, the exchange faced a paper work crises that resulted in four day weeks [1].

The automation of financial markets brought a required change to industry. One of the first steps towards automated markets was marked by the introduction of the computerised quote delivery systems in the 1960s [5]. The Quotron I, developed in 1960, was the first quote delivery system that allowed brokers to automatically request and retrieve prevailing stock prices [6]. The Quotron II and Ultronics' *Stockmarker* improved upon this invention by facilitating the transmission of market data (such real-time prices, and opening highs and lows) in digital format [6, 7].

However up until the late late 1960s most transactions were still executed manually and the data transmitted by quote delivery systems only included information concerning transactions [1][6–8]. The *Institutional Network Corporation* (Instinet) is credited with launching the first automated trading platform in history. The platform, introduced in December 1969, enabled a network U.S institutional investors to submit limit orders to the network, and automatically trade with each other [9, 10]. Further, the platform allowed the investors to view the bid and ask prices for any given security in the network[10]. This technology was ground-breaking because — in addition to facilitating autonomous trades — it allowed for the efficient collection of financial markets data. Instinet's innovation was followed by the establishment of the National Association of Securities Dealers Automated Quotations (NASDAQ) in 1971. The introduction of NASDAQ was subsequently followed by the migration from open outcry to electronic trading in several markets across the world [5]. As computerisation and electronic trading facilitated data collection, the decrease in data storage costs increased the availability of this data to both market participants and researchers. Today, several data vendors offer tick-by-tick, time-stamped

[1]The information of orders that did not result in transaction — like the prices and volumes of unmatched bid and ask orders — was generally not recorded.

transactions and quotes data. Thompson Reuters provides global tick data across all asset classes and more than 400 stock exchanges; the daily Trades and Quotes (TAQ) database provides access to all trades and quotes for all issues traded on NYSE, NASDAQ, and Regionals from 1993 to present [11, 12].

These data sets — referred to as *tick data* — are an increasingly significant resource because, among other things, they can be used to:

- Monitor the "health" today's, fast paced, ever changing markets [13].
- Drive data driven innovation and policy formation [14].
- Validate scientific paradigms for intraday market dynamics [8, 15].
- Back test trading strategies and estimate pre- and post-trade execution costs [16].

However, tick data sets are often large in volume, inhomogeneous in structure and contaminated with erroneous data. Therefore, attempting to conduct tick data intensive studies can prove to be a difficult task. Thus, this dissertation aims to provide a data processing procedure that can be used to efficiently conduct large scale tick data driven research.
Several studies in the literature have used and/or discussed tick data sets to analyse various aspects of financial markets. Engle (1996)[17] coined to term *ultra-high frequency data* to describe the comprehensive nature of tick data sets. The author applies the Autoregressive Conditional Duration model to IBM transaction data to develop estimates and measures of conditional variance. Goodhart and O'Hara (1997)[8] discusses issues relating to the analysis and application of tick data data sets. Lillo *et al* (2003)[18] uses data from the TAQ database to study the price impact of trades for 1000 New York Stock Exchanges (NYSE) traded stocks. Zivot (2005)[15] uses S-PLUS (a commercial implantation of the *S* programming language [19]) to analyse tick data. Giampaoli (2009)[20] applies methods from Fourier analysis to study tick data, and J. Glattfelder *et al* (2010)[21] discovered 12 scaling laws from five years worth of tick data for 13 exchange rates. Du Preez *et al* (2015)[22] used tick data from the Johannesburg Stock Exchange (JSE) and BM&FBOVESPA Securities, Commodities and Futures Exchange (now Brasile Bolsa Balcão or B3) to investigate the existence of stylized facts in these stock exchanges. Harvey *et al* 2017 [23] uses tick data from the Thompson Reuters History Tick database to investigate the effects of a fee restructuring on the JSE. (We review some of these studies in detail below.)

Although tick data sets have been extensively used in the literature, authors do not usually disclose the procedure that they used to manage their data and prepare it for analysis. This dissertation aims to contribute to previous studies by providing procedures (or a *workflow*) to manage large data ticks for empirical analysis. The workflow that we describe here is for a particular sample of tick data, however, since tick data sets generally have the same structure, we hope that this workflow can easily be adapted for different sets of tick data. The secondary goal of this dissertation is to demonstrate that this worklow is viable. We do this by investigating the empirical *price response* of trades from a one terabyte tick data set.

1.1 Price response and the limit order book

In the continuous trading session of order driven markets, participants can trade directly with each other (through a broker) by submitting *limit orders* and/or *market orders*. A limit order is a direction given to a broker to buy or sell a specific quantity of a security at a given price or better (e.g buy 100 shares of stock *xyz* at a price no more than R50 per share, or sell 20 shares of stock *abc* at a price no less than R70 per share)[24]. Buy limit orders are referred to as *bids to buy* and sell limits orders are called *asks* or *offers to sell*. Limit orders are used by market participants that wish to lock in on a particular price. On the other hand, participants that wish to transact immediately can do so by feeding on active limit orders using *market orders*. A market order is similar to a limit order, however, it does not specify a limit price (e.g buy 50 shares of stock *wxy*). Generally, buy market orders (or *buyer initiated trades*) are filled by transacting at the lowest ask price, with the earliest timestamp, and sell market orders (or *seller initiated trades*) are filled by transacting at the highest bid with the earliest timestamp. Put differently, limit orders are executed on a price-time priority. Although market orders provide immediacy, they come with a risk that we shall refer to as *price response*. Price response is the effect of a market order on subsequent prices. Buyer initiated trades tend to increase the market price, and seller initiated trades tend to decrease it. To see why this might occur, one needs to look no further than at the mechanics of continuous trading: since limit orders are generally executed by price-time priority, a market order that depletes the inventory available at the best price will transact at progressively worse prices as the order *walks* the limit order book (i.e the record of all active limit orders). Thus, price response can lead to unintended transaction and increase risk. Monitoring and controlling price response is therefore a very active area of research in trading firms and in market microstructure research communities [25].

Some of the most important question concerning price response are related to temporal dependence (do prices stay the same or revert back to their average levels after "responding" to trades?) and volume dependence (do larger trades induce a more significant price response?). On the question of the temporal dependence of price response, Kraus and Stoll (1972) [26] investigated whether the price movements accompanying block trades on the NYSE can be ascribed to changes in the underlying value of the traded asset (termed *information effect*) or temporary deviations in prices (termed *distributional effect*). The authors reported an information effect for plus tick block trades and a distributional effect for minus tick bock trades [2]. Touching on both the temporal and volume dependence of price response, the Kyle model, first introduced in Kyle (1985)[27], assumes that price response is linear in volume and permanent in time. In particular, the price adjustment rule Δp, over the interval Δt, of a market maker is governed by the equation

$$\Delta p = \lambda \epsilon v \tag{1.1}$$

Here λ is a measure of liquidity that is inversely proportional to the liquidity of the market and $v = |v_b - v_s|$, where v_b and v_s are the buy and sell volumes (respectively) over the interval Δt. Also, $\epsilon = 1$ if $v_b > v_s$ and $\epsilon = -1$ when $v_b < v_s$. Further, the price change between the interval $t = 0$ and $t = T = N\Delta t$, is modeled by

[2] A plus (minus) tick block trade is a trade that occurs at a price that is above (below) the previous transaction price.

$$p_T - p_0 = \sum_{n=0}^{N-1} \Delta p_n \tag{1.2}$$

$$= \lambda \sum_{n=0}^{N-1} \epsilon_n v_n \tag{1.3}$$

signifying permanent impact.

To capture the volume and time dependence of price response, Potters and Bouchaud (2003) [28] defined the *response function* $\mathcal{R}(\tau, V)$. This function measure the average mid-price variation between times t and $t+\tau$, conditional to transaction volume V. That is

$$\mathcal{R}(\tau, V) = \langle \varepsilon(t)(m(t+\tau) - m(t))|V \rangle \tag{1.4}$$

Here, $\varepsilon(t) = +1$ if the trade at time t is buyer initiated , and $\varepsilon(t) = -1$ if it is seller initiated. And $m(t)$ is the mid-price at time t. If t is expressed in terms of event time, where an *event* is a trade, then we can write the response function as:

$$\mathcal{R}(\ell, V) = \langle \varepsilon_n(m_{n+\ell} - m_n)|V_n = V \rangle \tag{1.5}$$

$\mathcal{R}(\ell, V)$ was first defined in Bouchaud *et al* (2003) [29] and it measures the price increase (decrease) at time ℓ (i.e, after ℓ trades) conditional on a buyer (seller) initiated trade, of volume V, at time 0 . Here ε_n [3] is the trade direction of the n^{th} trade, V_n is the volume of the n^{th} trade, and m_n is the mid-quote just before the n^{th} trade. Furthermore, the quantity given by $\mathcal{R}(\ell = 1, V)$, is referred to as *price impact* and has been the subject of many papers in market microstructure and econophysics literature. It is also one of the quantities that we seek to measure for our price response investigation.

1.2 Price Impact

Price impact refers to the relationship between the change in the mid-price (of an asset) subsequent to a trade of volume V. As suggested by the Kyle model above, the magnitude of price change following a trade of volume V is proportional to the traded volume. Hausman and Lo (1992)[30] used *ordered probit* — a statistical model for discrete random variables — to estimate the conditional distribution of trade-to-trade price changes from 1988 transaction data for 100 U.S traded stocks. In contrast to the Kyle model, the authors reported a non-linear relationship between price change Δp and traded volume V. Using transaction data for 37 large investment management firm, Chan and Lakonishok (1995)[31] found that price impact and execution cost are related to firm capitalization. Lillo *et al* (2002)[32] contributes to findings of Hausman and Lo (1992)[30] and Chan and Lakonishok (1995)[31] by considering transaction and quote data, from 1995 to 1998, of the 1000 most highly capitalized stocks on the New York Stock Exchange. The authors used the following procedure to calculate the price impact of each stock:

[3] $\varepsilon = +1$ for a buyer initiated trade, and $\varepsilon_n = -1$ for a seller initiated trade.

For each transaction of volume normalised volume ω [4] and at time t_i, when the next event in the (time stamped) data set is a quote revision, the change in the mid-price

$$\Delta p(t_{i+1}) = p(t_{i+1}) - p(t_i) \tag{1.6}$$

is computed. Here $p(t_i)$ is the log mid-price preceding the transaction and $p(t_{i+1})$ is the log mid-price following the transaction. When the event (following a transaction) is another transaction, the price change Δp is set to zero. The relationship between the average price shift and traded volume is then investigated. This is done separately for buyer and seller initiated trades. Here the algorithm developed by Lee and Ready (1991) [33] was used to distinguish buyer from seller initiated trades. To put all stocks on roughly the same footing, the transaction size of each trade was normalized by the average transaction in each year. Lastly, in order to investigate the average behaviour between price shift and transaction volume, the data was binned based on transaction size and the average price shift was calculated for each bin. After performing this procedure for a high capitalisation stock (General Electric Co) and a low capitalisation stock (Rectifier Corp), the authors found that the buyer initiated price impact curve of the high capitalisation stock increases as $\omega^{0.6}$, where ω is the normalised transaction size, and that of the lower capitalisation stock increases like $\omega^{0.5}$ for small ω, and $\omega^{0.2}$ for large ω. Further, the log-log plot of the price impact curve of each stock (figure 1 in Lillo *et al* (2002)[32]) indicates that the high capitalisation stock has a smaller average price shift than the low capitalisation stock, for all values the ω. To further investigate the relationship between market capitalisation and price impact, the authors grouped the 1000 considered stocks into 20 groups, ordered by market capitalisation. The number of stocks in each group was chosen to keep the number of transaction in each group roughly the same. The authors then used the procedure described above to calculate the average price change versus transaction size in each group. Plotting the calculated price impact curves on a double log plot, the authors found that the average price impact curves stratify themselves, from top to bottom, in increasing order of market capitalisation. To put the results in more mathematical terms, the authors performed a regression of the form

$$\Delta p = \frac{\text{sign}(\omega)|\omega|^{\beta}}{\lambda} \tag{1.7}$$

to fit each price impact curve. Here, Δp is the average price change, ω is the normalized transacted volume, β represents the slope of the price impact function and λ is some liquidity parameter. The authors found that the slope of each curve varies from 0.5 for small volumes in higher capitalisation stocks, to 0.2 for large volumes in lower capitalisation stocks. The authors also found that the liquidity parameter λ scales $C^{0.4}$, where C is the average market capitalisation of the considered stock group. The observations made were consistent for each considered year.

Taking a more theoretical approach to analysing price impact, Iori *et al* (2003) [34] introduced a model of double auction and used dimensional analysis, simulations and analytical approximations to make testable predictions of the price impact function in an order driven market. In agreement with empirical results in [30] and [32], the authors found that the price impact function is always concave. Lim and Coggins (2005) [35] performed a study similar to that of Lillo *et al* (2002) [18]. The authors considered intraday data of the top 300 stocks on the (Australian Stock Exchange) ASX. However,

[4] Here ω is calculated by normalising each transaction by the average value of the stock in question stock in each year [32].

as opposed to measuring the average price impact of buyer and seller initiated trades on the mid-prices, the authors investigated the average impact of buyer initiated trades on the ask price, and the average impact of seller initiated trades on the bid price. The price change for each trade (succeeded by a quote revision) of volume v was calculated as:

$$\Delta p_b(t_{i+1}) = a(t_{i+1}) - a(t_i) \tag{1.8}$$

for buyer initiated trades. Here, $a(t_i)$ is the logarithm of the ask price ahead of the trade with time stamp t_i, and $a(t_{i+1})$ is the logarithm of the ask price following the transaction. Similarly, the price change corresponding to seller initiated trades is calculated as

$$\Delta p_s(t_{i+1}) = b(t_{i+1}) - b(t_i) \tag{1.9}$$

Here, $b(t_i)$ is the logarithm of bid price before the trade, and $b(t_{i+1})$ is the logarithm of bid price after the trade. Furthermore, the traded volume v_{ij} for each trade j on day i, was normalized to ω_{ij}. Here,

$$\omega_{ij} = \frac{v_{ij}}{\sum_{m=1}^{T_i}} \left(\frac{N}{\sum_{d=1}^{N} T_d} \right)^{-1} \tag{1.10}$$

T_i represents the total number of trades on day i and N represents the total number of considered days. The authors grouped the considered stocks into 10 liquidity groups in such a way that the number of transactions in each group is roughly the same. As with Lillo *et al*, the authors found that stocks with a higher market capitalisation experienced a smaller magnitude of average price impact.

In addition to the United States of America (USA) and Australia, price impact has also been investigated in the South African equity market. Du Preez *et al* (2015) [22] conducted a price impact investigation for 5 JSE listed stocks: Anglo American PLC (AGL), Aspen Pharmacare Holdings Ltd (APN), BHP Billiton PLC (BIL), MTN Group Ltd (MTN), Standard Bank Group Ltd (SBK). In agreement with findings in [32] and [35], the authors reported price impact curves that increase (decrease) like a power law for buyer (seller) initiated trades. Harvey *et al* (2017) [23] used data from the Thompson Reuters Tick History Database to measure the price impact and estimate the master curves of JSE listed stocks using the procedures described in [32]. The authors considered the transactions and quotes data of the JSE Top 40 stocks for the period between January 01, 2013 to December 31, 2013. The authors used the calculated price impacts as a liquidity proxy to investigate the impact of structural changes to the market in relation to the changes in the pricing model on the JSE. The authors found that the change in the pricing model may have increased price impact for small transaction volumes.

On the invariance of price impact, Patzelts and Bouchaud (2017)[36] showed that price impact has a universal non-linear shape for trades aggregated on any intra-day. The authors found that the function varies very little across different instruments. Here, the considered instruments are US stocks, Nordic stocks and EUREX futures. On the other hand, Kyle and Obizhaeva(2018)[37] showed that a tightly parameterised universal market impact formula exists if one makes market microstructure in variance assumption (described in Kyle and Obizhaeva(2018)[37]). On the invariance of price impact, Patzelts and Bouchaud (2017)[36] showed that price impact has a universal non-linear shape for trades aggregated on any intra-day. The authors found that the function varies very little across different instruments. Here, the considered instruments are US stocks, Nordic

stocks and EUREX futures. On the other hand, Kyle and Obizhaeva(2018)[37] showed that a tightly parameterised universal market impact formula exists if one makes market microstructure in variance assumption (described in Kyle and Obizhaeva(2018)[37]). These studies are of interest to us because we consider and investigate the invariant properties of price response across different exchanges in this dissertation.

1.3 Time dependence of price response

As observed in the study by Kraus and Stoll (1972)[26], the price response to a trade can have a temporary effect or a long lasting/permanent effect. Almgren and Chriss (2000)[38] distinguishes between temporary and permanent price impact. The authors describe temporary price impact as "temporary imbalances in supply in demand caused by our trading leading to temporary price movements away from equilibrium" and permanent price impact as "changes in the 'equilibrium' price due to our trading, which remain at least for the life of our liquidation". Huberman and Stanzl (2004)[39] show that when the price impact of trades is permanent and time dependent, only linear price impact functions can rule out out no quasi-arbitrage.

Bouchaud *et al* (2003)[29] and Lillo and Farmer (2004) [40] independently reported highly persistent trade sign autocorrelation $C(\ell)$ for some stocks traded on the Paris Stock Exchange and the London Stock exchange. This means that buyer(seller) initiated trades are more likely to be followed by other buyer(seller) initiated trades for these stocks, even after a large number of trades. All else constant, if price response is permanent, since buyer(seller) initiated trades tend to drive the price upwards (downwards), one would expect long range autocorrelations in trade signs signs to lead correlated price changes. This is demonstrated in Lillo and Farmer (2004) [40]: the authors synthesize a price return series using real market, correlated trade signs and a deterministic price impact function of the form $\Delta p = \text{sign}(\omega)|\omega|^{\beta}/\lambda$ to show that buyer(seller) initiated trades tend to drive the price upward(downwards). However, both [29] and [40] argue that price returns are uncorrelated even though trade signs exhibit long range autocorrelations. To support their claim, [29] note that the dispersion of the price $\mathcal{D}(\ell) = \langle (m_{n+\ell} - m_n)^2 \rangle$, behaves as $\mathcal{D}(\ell) \approx D\ell$, even though trade signs are correlated. Here, m_n is the is the mid-price ahead of the n^{th} trade, and $m_{n+\ell}$ is the mid-price before the $(n+\ell)^{th}$ trade.

Thus, in order to reconcile the observed persistent trade sign autocorelations with non-correlated price changes, Bouchaud [29] introduced a model under which the mid-price p_n at time n is modulated by a non-linear propagator function, $G_0(\ell)$. In particular, the price process is governed by the equation

$$p_n = \sum_{n'<n} G_0(n-n')\varepsilon_{n'} \ln V_n + \sum_{n'<n} \eta_{n'} + \varepsilon_n \tag{1.11}$$

Here, ε_n is the sign of the n^{th} trade (+1 for a buyer initiated trade, and -1 for a sell), V_n is the traded volume of the n^{th} trade and η_n represents changes in the mid-price not attributed to changes in order flow. $G_0(\ell)$ is the average impact of a trade at time 0, after ℓ trades. Thus, $G_0(\ell)$ seeks to describe how the magnitude of impact of a single trade evolves over time. [29] notes that $G_0(\ell)$ has to decrease like a power law to allow for long range trade sign autocorrelations to coexist with with uncorrelated prices.

Looking at the price response function $\mathcal{R}(\ell, V)$, [29] and [28] reported that this function can be written as

$$\mathcal{R}(\ell, V) \approx \mathcal{R}(\ell) f(V) \tag{1.12}$$

for the considered set of data. Here, the authors reported that $f(V) \propto \ln V$, and $\mathcal{R}(\ell) = \langle \varepsilon_n (m_{n+\ell} - m_n) \rangle$ measures the magnitude of average mid-price increase (decrease) at time ℓ conditional on a buyer (seller) initiated trade at time 0 [29]. Using data from the Paris stock market, [29] report an initial increase in $\mathcal{R}(\ell)$ followed by a decrease for large values of ℓ. [29] notes that empirically, the function $G_0(\ell)$ and $\mathcal{R}(\ell)$ measure different quantities because trade signs ε_n exhibit long range correlations. Since $\mathcal{R}(\ell)$ is the coefficient of $f(V)$ in equation (1.12), we shall refer to $\mathcal{R}(\ell)$ as the *average price response coefficient*.

In this study, we mainly use six measured quantities to investigate price response: the price impact function $\mathcal{R}(\ell = 1, V)$ to investigate the volume dependence of price response, the autocorrelation $\mathcal{C}(\ell)$ of trade signs to investigate the nature of order flow, the dispersion of the price $\mathcal{D}(\ell)$ to investigate the degree of correlation in price changes and the bare response function $G_0(\ell)$ and price response coefficient $\mathcal{R}(\ell)$ to investigate the temporal dependence of price response. Our study also seeks to investigate if liquidity, as measured by average daily volume (ADV) has an effect on price response, as measured by $\mathcal{R}(\ell = 1, V)$, $\mathcal{C}(\ell)$, $\mathcal{D}(\ell)$, $\mathcal{R}(\ell)$ and $G_0(\ell)$.

1.4 Price response and liquidity

In addition to providing a data processing *workflow* that can be used to conduct tick data intensive research, this study aims to investigate the relationship between price response and liquidity for the markets and data described below. For a detailed consideration of liquidity, refer to [41] and [2]. In this dissertation liquidity is homogenised across a selection of emerging markets (discussed below). Here, we measure liquidity by average daily volume (ADV) and average daily traded value (ADTV). Further, we measure price response using the price impact function $\mathcal{R}(\ell = 1, V)$, the autocorrelation of trade signs $\mathcal{C}(\ell)$, the dispersion of the price $\mathcal{D}(\ell)$, the price response coefficient $\mathcal{R}(\ell)$ and the bare response function $G_0(\ell)$.

1.5 Considered data and markets

This study uses a tick dataset obtained from the Thompson Reuters Tick History (TRTH) database. The data spans from the 4^{th} of January 2010 to the 10^{th} of May 2016, and contains the quotes and trades of stocks listed in some of the largest stock exchanges in emerging markets. This includes the Bombay Stock Exchange (BSE), Brasil Bolsa Balcão (B3), the Egyptian Exchange (EGX),the Johannesburg Stock Exchange (JSE), the Moscow Exchange (MEOX), the Nairobi Securities Exchange (NSE) and the Shanghai Stock Exchange (SSE). This study is novel because the data is homogenised across the studied markets by the TRTH dataset. This means that results are directly comparable because the same data pre-processing is used.

1.6 Provided GitHub repositories

This document can be used in conjunctions with two GitHub repositories that we provide, *Calculate_PriceResponse* and *mongoDB_Tickstore*, to reproduce some of our results. *Calculate_PriceResponse* contains all of the MATLAB classes, functions and scripts that we

used to calculate price response for the considered data. Furthermore, it contains the data and MATLAB scripts required to reproduce most of the plots in this dissertation. The second repository, *mongoDB_Tickstore*, contains the necessary files required to reproduce a working version of the data processing worflow that we discuss in later chapters. Furthermore, the repository contains a sample of the TRTH data we used. The url of *Calculate_PriceResponse* is https://github.com/Telmakaza/Calculate_PriceResponse and that of *mongoDB_Tickstore* is https://github.com/Telmakaza/mongoDB_Tickstore.

1.7 Overview

The overview of this dissertation is as follows. Chapter 2 describes the procedure that we followed to prepare the considered data set for analysis. Furthermore, we highlight the method that we used to to calculate price response parameters (i.e $\mathcal{R}(\ell = 1, V)$, $\mathcal{C}(\ell)$, $\mathcal{D}(\ell)$, $\mathcal{R}(\ell)$ and $G_0(\ell)$) in this chapter. Chapters 3 uses the described methods and workflow to analyse the data and price response of trades for each of the considered market. Chapter 4 presents the conclusion. This document also includes appendices that supplement the material presented in this dissertation. Appendix A lists all of the considered stocks and their Industry Benchmark Classification (ICB). Appendix B provides a technical documentation for the key components of the developed data processing workflow. Appendix C provides the source codes that we used to construct the data processing workflow and analyse the data. Lastly, appendix D lists the specifications of all of the machines that we used in this project.

Chapter 2

Methodology

The project was aimed at studying and comparing price response across a selection of emerging markets in a manner that is both transparent and easy to replicate. Thus, this chapter presents the procedures that we used to develop a data processing worklfow for a tick data set. Although the procedures described here were designed for use with a particular data set, in particular, the Thomson-Reuters Tick History (TRTH) data-set that has been homogenised in like-for-like attributes and fields across markers, it can be used for similar tick data sets from other vendors and markets, e.g. Bloomberg [42]. The layout of this chapter is as follows. Section 2.1 describes the considered data set and section 2.2 discusses some of the considerations that one needs to make before working with this data. Section 2.3 describes the steps taken in building this the data processing workflow. Lastly, since this dissertation mainly uses this workflow to analyse the price response of trades, section 2.4 highlights the procedures that we used to calculate the price response parameters discussed in chapter 1.

2.1 Data description

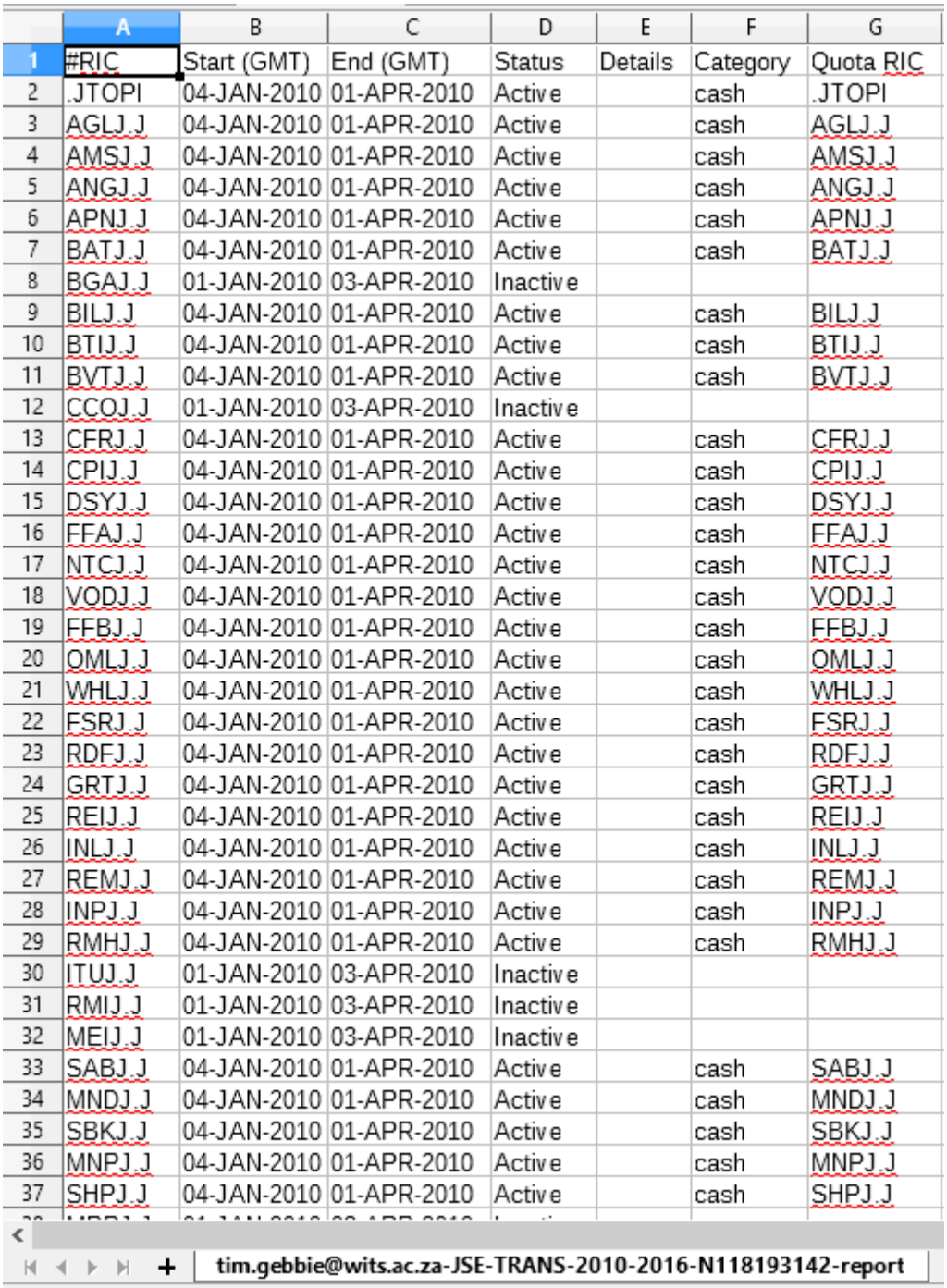
We considered data from stock markets in Brazil, India, Egypt, South Africa, Kenya, Russia and China. The data was obtained from Thompson Reuters and spans from January 04, 2010 to May 10, 2016. The list of all of the considered stocks can be found in appendix A. The data is stored in the form of comma delimited files with the extention `.csv`. Each csv is approximately six gigabytes in size. We shall refer to these data files as *transactions csv files*. There is a total of 189 *transactions csv* files, making the total size of the considered data approximately one terabyte. Each file contains multiple rows and columns. Rows represent *top-of-the-book* events (i.e quote updates and trades) and each column describe the characteristics of each market event. Figure 2.1 provides a screen shot of one of the *transactions csv* file for the JSE. Table 2.1 describes all of the headings of the considered column in these files. As can be seen from figure 2.1, the considered data has the characteristics of semi-structured data. That is, some rows have missing attributes and others have more attributes than others [43]. Each transactions csv file can contain the data of up to 30 different stocks from January 04, 2010 up to May 10, 2016. In addition to this, the data also contains *report csv* files. These files list all of the consider stock in each market. Thus, *report csv* files were used as the key input in producing the tables listed in appendix A, and analysing the data of each market. Except for *Brasil Bolsa Balcão* (B3), each market has one *report csv* file. The B3 has three. Figure 2.2 provides a screen shot of the *report csv* for the JSE data.

Date[G]	Time[G]	GMT Offset	Type	Price	Volume	Market VWAP	Bid Price	Bid Size	Ask Price	Ask Size
16-Jul-15	06:01:53	5.5	Quote					37		
16-Jul-15	06:01:54	5.5	Quote					47		
16-Jul-15	06:01:55	5.5	Quote					37		
16-Jul-15	06:01:58	5.5	Trade	2529.65	3	2539.5411				
16-Jul-15	06:01:59	5.5	Trade	2529.65	3	2539.1169				
16-Jul-15	06:02:03	5.5	Trade	2529.65	18	2542.1372				
16-Jul-15	06:02:03	5.5	Trade	2529.65	3	2541.713				
16-Jul-15	06:02:03	5.5	Quote							1
16-Jul-15	06:02:04	5.5	Trade	2529.65	1	2541.5717				
16-Jul-15	06:02:04	5.5	Quote							2
16-Jul-15	06:02:05	5.5	Trade	2529.65	14	2539.5943				
16-Jul-15	06:02:05	5.5	Quote				2529.65		2531.3	20
16-Jul-15	06:02:06	5.5	Quote				2530	12		
16-Jul-15	06:02:08	5.5	Quote				2529.75	2		
16-Jul-15	06:02:08	5.5	Quote					27		
16-Jul-15	06:02:09	5.5	Quote				2530	12		
16-Jul-15	06:02:09	5.5	Quote				2530.35	2	2531.5	31
16-Jul-15	06:02:10	5.5	Quote				2529.85	30	2531.45	68
16-Jul-15	06:02:11	5.5	Quote				2529.95	2		
16-Jul-15	06:02:11	5.5	Quote				2530			
16-Jul-15	06:02:12	5.5	Quote				2530.2			
16-Jul-15	06:02:12	5.5	Quote				2530.25			
16-Jul-15	06:02:12	5.5	Quote				2530.3			
16-Jul-15	06:02:13	5.5	Quote					27		

Figure 2.1: Screen shot of one of the comma delimited `.csv` files for the considered Johannesburg Stock Exchange data set. Each row in this file represents a market event (e.g quote update, transaction etc) and columns discribe the attributes of market events.

Data file attribute	Description
`#RIC`	Reuters instrument code. It identifies financial instruments and indices
`Date[G]`	GMT date string of market event in *DD-MMM-YYYY* format
`Time[G]`	GMT time string in *hh:mm:ss.000* format.
`GMT Offset`	The GMT offset
`Type`	The type of an event. There are 4 event types: auctions, quotes, trades and market conditions.
`Price`	The price of a trade.
`Volume`	The volume of a trade.
`Bid Price`	The prevailing best bid price.
`Bid Size`	The volume of the prevailing best bid price.
`Ask Price`	The prevailing best ask price.
`Ask Size`	The volume of the prevailing best ask price.

Table 2.1: Describtions the attributes in each data file. The attributes charecterise the features of each market event or *row* in every data file. For example, a market event that has the an *attribute-value* pair list given by "`RIC:BGAJ.J, Date[G]:04-JAN-2010, Time[G]:15:33:63.785, Type:Trade, Price:201, Volume:400`" describes a trade for the stock with the ticker *BGAJ.J* at 15:33:63.785 on the 4^{th} of January 2010. Furthermore, this attribute list tells use that the price of this trade was 201 rands per share and the total number of shares transacted at that time was 400.



	A	B	C	D	E	F	G
1	#RIC	Start (GMT)	End (GMT)	Status	Details	Category	Quota RIC
2	.JTOPI	04-JAN-2010	01-APR-2010	Active		cash	.JTOPI
3	AGLJ.J	04-JAN-2010	01-APR-2010	Active		cash	AGLJ.J
4	AMSJ.J	04-JAN-2010	01-APR-2010	Active		cash	AMSJ.J
5	ANGJ.J	04-JAN-2010	01-APR-2010	Active		cash	ANGJ.J
6	APNJ.J	04-JAN-2010	01-APR-2010	Active		cash	APNJ.J
7	BATJ.J	04-JAN-2010	01-APR-2010	Active		cash	BATJ.J
8	BGAJ.J	01-JAN-2010	03-APR-2010	Inactive			
9	BILJ.J	04-JAN-2010	01-APR-2010	Active		cash	BILJ.J
10	BTIJ.J	04-JAN-2010	01-APR-2010	Active		cash	BTIJ.J
11	BVTJ.J	04-JAN-2010	01-APR-2010	Active		cash	BVTJ.J
12	CCOJ.J	01-JAN-2010	03-APR-2010	Inactive			
13	CFRJ.J	04-JAN-2010	01-APR-2010	Active		cash	CFRJ.J
14	CPIJ.J	04-JAN-2010	01-APR-2010	Active		cash	CPIJ.J
15	DSYJ.J	04-JAN-2010	01-APR-2010	Active		cash	DSYJ.J
16	FFAJ.J	04-JAN-2010	01-APR-2010	Active		cash	FFAJ.J
17	NTCJ.J	04-JAN-2010	01-APR-2010	Active		cash	NTCJ.J
18	VODJ.J	04-JAN-2010	01-APR-2010	Active		cash	VODJ.J
19	FFBJ.J	04-JAN-2010	01-APR-2010	Active		cash	FFBJ.J
20	OMLJ.J	04-JAN-2010	01-APR-2010	Active		cash	OMLJ.J
21	WHLJ.J	04-JAN-2010	01-APR-2010	Active		cash	WHLJ.J
22	FSRJ.J	04-JAN-2010	01-APR-2010	Active		cash	FSRJ.J
23	RDFJ.J	04-JAN-2010	01-APR-2010	Active		cash	RDFJ.J
24	GRTJ.J	04-JAN-2010	01-APR-2010	Active		cash	GRTJ.J
25	REIJ.J	04-JAN-2010	01-APR-2010	Active		cash	REIJ.J
26	INLJ.J	04-JAN-2010	01-APR-2010	Active		cash	INLJ.J
27	REMJ.J	04-JAN-2010	01-APR-2010	Active		cash	REMJ.J
28	INPJ.J	04-JAN-2010	01-APR-2010	Active		cash	INPJ.J
29	RMHJ.J	04-JAN-2010	01-APR-2010	Active		cash	RMHJ.J
30	ITUJ.J	01-JAN-2010	03-APR-2010	Inactive			
31	RMIJ.J	01-JAN-2010	03-APR-2010	Inactive			
32	MEIJ.J	01-JAN-2010	03-APR-2010	Inactive			
33	SABJ.J	04-JAN-2010	01-APR-2010	Active		cash	SABJ.J
34	MNDJ.J	04-JAN-2010	01-APR-2010	Active		cash	MNDJ.J
35	SBKJ.J	04-JAN-2010	01-APR-2010	Active		cash	SBKJ.J
36	MNPJ.J	04-JAN-2010	01-APR-2010	Active		cash	MNPJ.J
37	SHPJ.J	04-JAN-2010	01-APR-2010	Active		cash	SHPJ.J

tim.gebbie@wits.ac.za-JSE-TRANS-2010-2016-N118193142-report

Figure 2.2: Screen shot of one of the comma delimited *report csv* file for the JSE data. These files list all of the stocks contained in all of the transactions csv files for each market. Each market has at least one *report csv* file.

2.2 Pre-analysis considerations

One of the main considerations to make when working with tick data is that these data sets are relatively large in size. Our data set consisted of consists of 189 `.csv` files, and each `.csv` is approximate six gigabytes in size. Loading one, six gigabyte csv file directly into a scientific computing environment such as MATLAB can prove to be a very resource intensive task that may be futile in the end. An easy way get around this problem is to stream the data using progressive loading techniques such as MATLAB tall arrays or Python Pandas. Although these tools are relatively easy to implement, they can become difficult to work with as the data size increases. Therefore, the workflow that we developed for this project was designed to be scalable.

2.3 Data processing workflow: development

The data processing workflow is made up of two main components: a database management system (DBMS), and a scientific computing programming language. In particular, we used MongoDB as the DBMS and MATLAB as the computational solution to carry out the data data analysis. MongoDB is a NoSQL, open source DBMS that is highly scalable and has a flexible data storage format that facilitates the efficient storage of semi-structured data sets such as tick data. One the other hand, MATLAB is a scientific computing programming language that is easy to use and has powerful tools, such as the *Parallel Computing Toolbox*, that expedite the process of analysing large chunks of data. Thus, developing the data processing workflow involved two main steps: (i) Building a database for the transactions csv files, and (ii) Building an application programming interface (API) that allows MATLAB to access the data inside the database.

(i) Building the database

The first step in building the tick database was to upload the data into a MongoDB database. We considered two different approaches for doing this. The first approach uses `mongoimport` (an application that included in the MongoDB setup) to upload the contents of the transactions csv files into a collection in the database. Since `mongoimport` can only upload one csv file at a time, we used MATLAB to iteratively upload the raw data into the database. The MATLAB script that we used is provided in appendix C.1. The advantage of this approach is that it is relatively easy to implement. However, its main drawback is that it does not give one the flexibility to choose the format in which the data is imported into the database. This is a significant setback as it can cause the database to take up more disk space and it can also somewhat limit ones ability to query data. This is precisely what happened with the our initial implementation of our database: dates and timestamps were stored as *Strings*, as opposed to more suitable and efficient MongoDB types such as *Dates* and *Timestamps*. Not only did this cost more disc space, it only allowed us to query data at daily times scales. Thus, an alternative data importation method was considered for comparison. This approach used C++ to convert the dates and times into a format that is (arguably) more *digestible* to MongoDB before import the data into the database. In the conversion, the values of the `Date[G]` attribute were turned into a *yyyy-mm-dd* format, and then the corresponding values of the `Time[G]` and `GMTOffset` attributes were concatenated to the newly formed values of `Date[G]`

to form values that have the format *yyyy-mm-ddTHH:MM:SS.FFF+0GMTOffset00*. These newly formed values were stored under a new attribute called `DateTime`, and the old values of the `Date[G]`, `Time[G]` and `GMTOffset` were discarded. We then imported the resultant file using `mongoimport` into the database. The values of the `DateTime` attribute are stored in a MongoDB `Date` object. This allows us to query data accurately and more efficiently—down to the last microsecond. The main disadvantage of this approach is that it is not as easy as the first to implement. The source code for the implementation is provided in appendix C.3. Further, a more technical discussion of the database construction is provided in the technical documentation in appendix B.

After all of the data was imported into the database (either using the first or second data import method), we built indexes for the values of the `#RIC` and `Date[G]/DateTime` attributes for the first implementation of the database, and did the same for the values of the `RIC` and `DateTime` attributes for the second implementation of the database. The purpose of doing this is to speed up the rate at which we can query data.

The final step in building the database was to enable user authentication to set up access control for our database. The procure for doing this is provided in the technical documentation in appendix B. Enabling user authentication is necessary because of the proprietary nature of the considered data set. T. Gebbie notes that "this is a necessary requirement to ensure that data retrieval and processing, and the associated log-files that evidence data usage and access, are always linked to specific individual user credentials, and thus specific designated users. This addresses the typical data-usage requirements of various data-vendors and necessarily links individuals users, to their credentials, to their responsibilities associated with appropriate data-usage, security and management"[42].

(ii) Developing a data access API

After building the database, we need to allow the computational solution, MATLAB, access to the data in the database [42, 44]. This requires the use of an appropriate interface that allows data queries to be passed from MongoDB to a MATLAB workspace. To this end, we developed an application programming interface API that instructs the database to extract the queried data from the database and save it to the file system on the client machine. Once the data has been exported to the client machine's file system, we use MATLAB's `textscan` function to import it into a MATLAB workspace. Each database implementation has a different data extraction API. We provide both of our APIs in appendix C.2 and C.4.

The MongoDB database and data extraction API are used in conjunction to create a scalable data processing workflow that can be used to conduct data intensive market microstructure research. Figure 2.3 illustrates a flow diagram of the developed workflow.

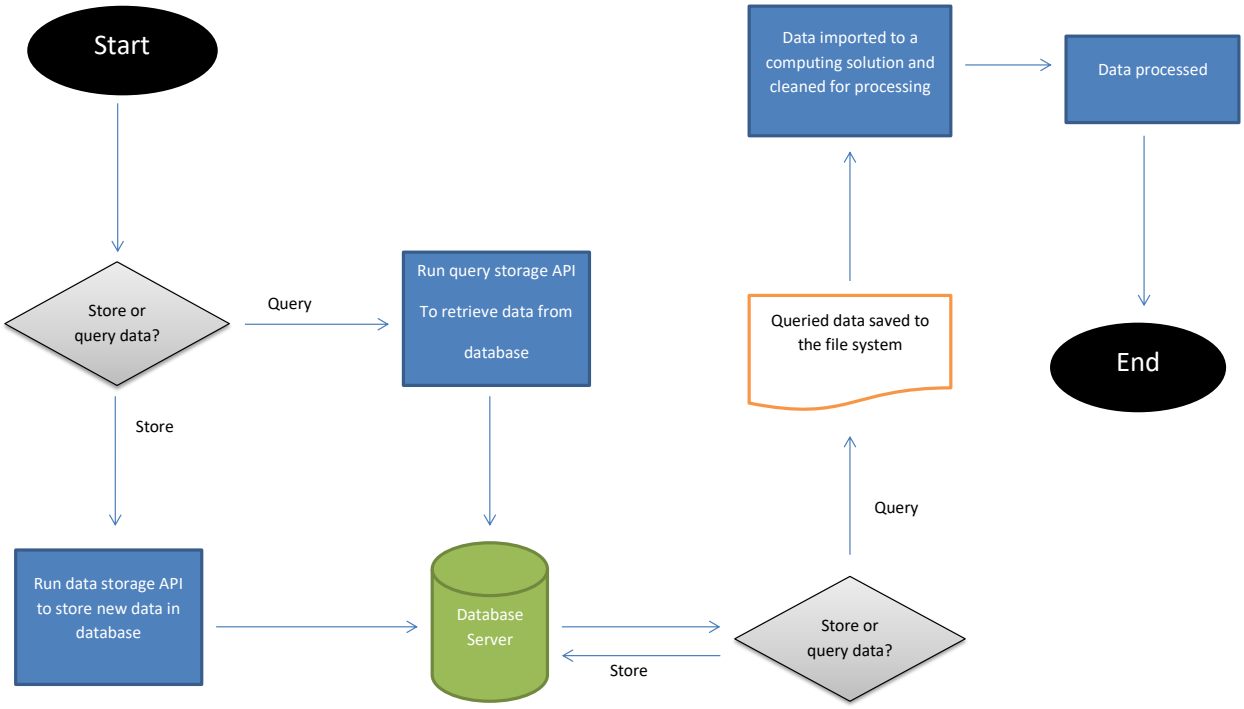


Figure 2.3: Flow diagram of proposed data processing workflow.

Exchange	Considered period (local time)	Considered period (UTC Time)
B3	10:10 - 16:45	12:10 - 18:45
BSE	09:25 - 15:20	03:55 - 09:50
EGX	10:10 - 14:20	08:10 - 12:20
JSE	09:10 - 16:50	07:10 - 14:50
NSE	09:40 - 14:50	06:40 - 11:50
MOEX	10:10 - 18:29	07:10 - 15:29
SSE	09:30 - 11:30 and 13:00 15:00	01:40 - 03:20 and 05:10 to 06:50

Table 2.2: Considered time period in local and UTC time for each exchange.

2.4 Calculating price response

Although the data processing workflow described in the previous section can be used for a variety of research projects, we used it to analyse the price response of trades for markets listed in chapter 1.5, using the data described in section 2.1 of this chapter. Price response was discussed in more detail chapter 1.1. Here, we present the formulas and methods that we used to estimate price response parameters. In particular, we highlight the methods that we used to calculate average price impact $\mathcal{R}(\ell = 1, V)$, trade sign autocorrelations $\mathcal{C}(\ell)$, price dispersion $\mathcal{D}(\ell)$, average price response coefficient $\mathcal{R}(\ell)$ and bare response $G_0(\ell)$. The method that we use to analyse price response is based on procedures used by [29,32,35]. In particular, from [32] and [35], we adopt the approach of considering the price response of single stocks, and then that of groups of stock, binned by number of transactions. From [29], we use the price response function coefficient $\mathcal{R}(\ell)$, the autocorrelation of trade signs $\mathcal{C}(\ell)$, the dispersion of the price $\mathcal{D}(\ell)$ and the bare response function $G_0(\ell)$ to investigate the temporal behaviour of price response.

2.4.1 Data cleaning

Before proceeding with any calculations, we started by *cleaning* the data[1]. In addition to transactions and quote updates, the data contained auctions. Our investigation on price response focuses on the effect of trades on subsequent mid-prices in continuous trading. Thus, the first step in data cleaning was to remove all auctions and events that occurred outside of continuous trading. Furthermore, the more volatile 10 minutes after opening was removed from the data. Futures close-outs were not considered in the data cleaning algorithm. Thus, the data associated with these periods was not necessarily removed.

We would like to highlight that the time stamps on the the data are in UTC time. Thus, any data cleaning that involved timestamps took this into account. As an example, the JSE continuous trading period starts from 09:00 and ends at 16:50 (South African Standard Time (SAST)) [45]. Thus, effectively, we would like to consider events that occur between 09:10 and 16:50 SAST. Since SAST = UTC + 2, we have UTC = SAST - 2. Therefore continuous trading period on the JSE starts at 07:00 and ends at 14:50 in UTC time. Thus, our data cleaning algorithm cleaned all events outside of the period from 07:10 to 14:50 UTC. Table 2.2 highlights the considered time period for each exchange in local and UTC time. Note that we did not take into account day light savings when isolating the data for each market.

[1] All of the steps described here are implemented in a the MATLAB script provided in appendix C.5

After removing non-continuous trading events from the data, we filtered out the quotes and trades of each stock as follows:

(i) For each sequence of quotes of quotes that have the same time stamp, we removed all of the quotes in this sequence, except for the one with the latest time stamp.

(ii) For each sequence of trades that have the same time stamp, we aggregated these sequence into single trades. The volume V of these aggregated trades was calculated by taking the sum of all of the traded volumes of each trade in the sequence. The price P of these trades was calculated as

$$P = \frac{\sum_{i=1}^{N} p_i v_i}{V}. \tag{2.1}$$

Here, p_i and v_i are (respectively) the price and volume of the i^{th} trade in the sequence. N is the total number of trades in the sequence. To conclude the data cleaning process, we removed all quotes and trades that have a price and/or volume of 0. The MATLAB script provided in appendix C.5 implements the data cleaning algorithm described above.

2.4.2 Individual stock analysis

We start by considering price impact on a single stock basis. After cleaning the data as decribed above, we classified trades as either being *buyer initiated* or *seller initiated*. To do this we use a variant [2] of the trade classification approach of Lee and Ready (2001)[33]. In particular, if the price of a trade is higher (lower) than the prevailing mid-quote, we classify it as a buyer (seller) initiated trade. We refer to these rules as the *Quote Rules*. Trades that cannot be classified by the Quote Rules were filtered out and classified according to the *Tick Rules*. The Tick Rules states that a trade that occurs on an up-tick (down-tick) or zero up-tick (down-tick) is buyer (seller) initiated. A trade is said to have occurred at an up-tick (down-tick) if the price of the trade is higher (lower) than the price of the previous trade; A trade is said to have occurred at a zero up-tick (down-tick) if the price of the trade is the same as that of the previous trade — which occurred on an up-tick (down-tick). Thus, after running the data through this trade classification algorithm, each classified trade had a corresponding trade signs. Buyer initiated trades were represented by +1 and seller initiated trades by -1. Trades that remained unclassified after applying the both the Quote and Tick Rule were removed from the data-set.

After trades were classified, we used methods in Lillo *et al* (2002)[32] and Lin and Coggins (2005)[35] to calculate price impact. In particular, for each transaction of volume v_n, if the event following the transaction was a quote update, we calculated the price shift Δp as:

$$\Delta p = m_{n+1} - m_n. \tag{2.2}$$

Here, m_n is the logarithm of the mid-quote just before the transaction and m_{n+1} is log mid-quote after the transaction. Thus each classified trade of volume v_n, is associated with a trade sign, ε_n and price change Δp_n. In order to make comparisons between

[2] Our trade classificaltion algorithm is different to that of Lee and Ready (2001)[33] because for each trade, the latter authors considered the quote that is in effect 5 seconds earlier than the trade; we consider the quote immediately before the trade—regardless of it's age in relation to the trade.

different stock meaningful, we used the method described in Lim and Coggins (2005)[35] to normalize the traded volume of each transaction. Let v_{ij} denote the traded volume of the j^{th} trade on the i^{th} trading day. v_{ij} was normalized to ωij. Here,

$$\omega_{ij} = \frac{v_{ij}}{\sum_{m=1}^{T_i} v_{im}} \left(\frac{N}{\sum_{d}^{N} T_d} \right)^{-1}, \tag{2.3}$$

where N denotes the total number of trading days (for example, $N \approx 251$ in one year) and T_i is the total number of trades on day i. After normalizing the traded volumes we create 20 logarithmically spaced *bins* from 10^{-2} to $10^{0.5}$, thus following the convention (but not the exact bin ranges) described in Harvey *et al* (2017)[23]. Here, we chose this specific bin range to attempt to increase the visibility and perceptibility of visually represented results. Bins were used to group trades by their normalized traded volume. Lastly, in order to investigate the relationship between transacted volume and price change, we calculated the median normalized transacted volume and the average price change in each bin. We did this separately for buyer and seller initiated trades for each stock. This was done for the years from January 2010 to May 2016. The results were then visualized on a log-log plot. (see figure 3.2.)

Next, we considered the the autocorrelation of trade signs, $\mathcal{C}(\ell)$, the normalized mean squared displacement of the price $\sqrt{\mathcal{D}(\ell)/\ell}$ the price response coefficient $\mathcal{R}(\ell)$ and the bare impact function $G_0(\ell)$ of trades for each stock. According to [29], the autocorrelation of trade signs is defined as:

$$\mathcal{C}(\ell) = \langle \varepsilon_{n+\ell} \varepsilon_n \rangle + \langle \varepsilon_n \rangle^2 \tag{2.4}$$

Here, ε_n is defined as above (that is, $\varepsilon_n = +1$ for a buyer initiated trade and $\varepsilon_n = -1$ for a seller initiated trade). In our investigation, we estimated $\mathcal{C}_0(\ell)$ by $\hat{\mathcal{C}}_0(\ell)$, where

$$\hat{\mathcal{C}}_0(\ell) = \frac{1}{M} \sum_{i=1}^{M-\ell} \varepsilon_{i+\ell} \varepsilon_i - \left(\frac{1}{M} \sum_{i=1}^{M-\ell} \varepsilon_i \right)^2 \tag{2.5}$$

Here, M is the total number of considered trades signs over a certain period. $\hat{\mathcal{C}}_0(\ell)$ was calculated for each trading day and averaged out across all of the trading days. As with the price impacts, we considered all the trading days from the 4^{th} of January 2010 to the 10^{th} of May 2016. When calculating $\mathcal{C}_0(\ell)$ for each stock, we set $M = 1000$ if the average number of trades per day for that stock is greater than or equal to 1000. Otherwise, we set M to be equal to the median number of trades per day for the stock in question.

Similarly, $\mathcal{R}(\ell)$, defined as $\mathcal{R}(\ell) = \langle \varepsilon_n (m_{n+\ell} - m_n) \rangle$, where m_n is the mid-quote before the n^{th} trade, was estimated by:

$$\hat{\mathcal{R}}(\ell) = \frac{1}{M} \sum_{i=1}^{M-\ell} \varepsilon_i (m_{i+\ell} - m_i) \tag{2.6}$$

and $\mathcal{D}(\ell)$, defined by $\mathcal{D}(\ell) = \langle (m_{n+\ell} - m_n)^2 \rangle$, was estimated by

$$\hat{\mathcal{D}}(\ell) = \frac{1}{M} \sum_{i=1}^{M-\ell} (m_{i+\ell} - m_i)^2 \tag{2.7}$$

Lastly, in order to calculate $G_0(\ell)$, we suppose that the mid-price at time n can be modelled by the equation:

$$m_n = \sum_j G_0(n-j)\varepsilon_j \ln V_j + \sum_j \eta_j \tag{2.8}$$

Substituting m_n into $\mathcal{R}(\ell) = \langle \varepsilon_n (m_{n+\ell} - m_n) \rangle$ yields:

$$\mathcal{R}(\ell) = \langle \ln V \rangle G_0(\ell) + \sum_{0<j<\ell} G_0(\ell-j)\mathcal{C}_1(j) + \sum_{j>0} \left[G_0(\ell+j) - G_0(j) \right] \mathcal{C}_1(j) \tag{2.9}$$

Here $\mathcal{C}_1(\ell) = \langle \varepsilon_{n+\ell}\varepsilon_n \ln V_n \rangle$. If we assume that the bare impact function $G_0(\ell)$ decays over time, we can write equation 2.9 as

$$\mathcal{R}(\ell) \approx \langle \ln V \rangle G_0(\ell) + \sum_{0<j<\ell} G_0(\ell-j)\mathcal{C}_1(\ell) \tag{2.10}$$

This can be written in matrix form as

$$\begin{bmatrix} \mathcal{R}(1) \\ \mathcal{R}(2) \\ \mathcal{R}(3) \\ \vdots \\ \mathcal{R}(n) \end{bmatrix} = \begin{bmatrix} \langle \ln V \rangle & 0 & 0 & \dots & 0 \\ \mathcal{C}_1(1) & \langle \ln V \rangle & 0 & \dots & 0 \\ \mathcal{C}_1(2) & \mathcal{C}_1(1) & \langle \ln V \rangle & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \mathcal{C}_1(n) & \mathcal{C}_1(n-1) & \dots & \mathcal{C}_1(1) & \langle \ln V \rangle \end{bmatrix} \begin{bmatrix} G_0(1) \\ G_0(2) \\ G_0(3) \\ \vdots \\ G_0(n) \end{bmatrix} \tag{2.11}$$

We solved for $G_0(\ell)$ by inverting the matrix on the right hand side of equation (2.11). For the data set that we considered, we found that this matrix is always invertible.

As with $\mathcal{C}(\ell)$, the estimates of $\mathcal{D}(\ell)$, $\mathcal{R}(\ell)$ and $G_0(\ell)$ were calculated separately for each trade day, and averaged out across all of the trading days to arrive at the final estimates.

2.4.3 Grouped stocks analysis

In order to investigate aggregate behaviour, we grouped the considered stocks into three different groups, order by liquidity. This was done separately for each stock exchange. The groups, labelled *Group 1* to 3 respectively, were created in such a way that the number of trades in each group is roughly the same. (The code pattern for doing this can be found in appendix C.12.) This was done separately for each year from January 2010 to May 2016. We then used the method described in section 2.4.2 to calculate the price impact of each group. The price impacts were calculated separately for each year, and then averaged out across all of the considered years.

We also estimated the autocorrelation of trade signs $\mathcal{C}(\ell)$, the price response coefficient $\mathcal{R}(\ell)$, the price dispersion $\mathcal{D}(\ell)$ and the bare response function $G_0(\ell)$ for each group. This was done using the methods described in section 2.4.2 with some minor adjustments. In

particular, the calculation for $\mathcal{R}(\ell)$ and $\mathcal{D}(\ell)$ were made dimensionless. That is, $\mathcal{R}(\ell)$ was estimated by

$$\hat{\mathcal{R}}(\ell) = \frac{1}{M}\sum_{i=1}^{M-\ell} \varepsilon_i \ln\left(\frac{m_{i+\ell}}{m_i}\right). \tag{2.12}$$

The price dispersion function $\mathcal{D}(\ell)$ was estimated by

$$\hat{\mathcal{D}}(\ell) = \frac{1}{M}\sum_{i=1}^{M-\ell} \ln^2\left(\frac{m_{i+1}}{m_i}\right). \tag{2.13}$$

Thus $G_0(\ell)$ was calculated by substituting equation (2.12) into (2.11). In order to make quantitative comparisons between different stock groups, we fit a linear regression of the form:

$$\ln \Delta p = \alpha \ln \omega - \ln \lambda. \tag{2.14}$$

to all of the measured price impacts $\mathcal{R}(\ell = 1, V)$. Here, we solved the optimization problem:

$$\min_{\alpha,\lambda \in \mathbb{R}} \{\ln \Delta p - \alpha \ln \omega - \ln \lambda\}. \tag{2.15}$$

Note that equation 2.14 comes from the fact that [32] fit an equation of the form $\Delta p = \text{sign}(\omega)|\omega|^{\alpha}$ to their results in order to make quantitative comparisons. Thus, we use their procedure in order to quantitatively compare the results that we obtained across different stock groups.

Lastly, we fit a relationship of the form

$$G_0(\ell) = \frac{\Gamma_0}{(\ell_0 + \ell)^{\beta/2}}. \tag{2.16}$$

to all of the measured bare impacts $G_0(\ell)$. We found Γ_0, ℓ_0 and β that satisfy

$$\arg \min_{\Gamma_0,\ell_0,\beta \in \mathbb{R}} \left\{ G_0(\ell) - \frac{\Gamma_0}{(\ell_0 + \ell)^{\beta/2}} \right\}. \tag{2.17}$$

and $\mathcal{C}_0$ and γ that satisfy

$$\arg \min_{\mathcal{C}_0,\gamma \in \mathbb{R}} \{\mathcal{C}(\ell) - \mathcal{C}_0 \ell^{\gamma}\} \tag{2.18}$$

See figures 3.7 and 3.10 for examples of some of the plots for grouped stocks.

2.4.4 Price response software class definitions

We designed and implemented two MATLAB software classes: `PriceResponse` and `GroupedPriceResponse` to implement and manage the analysis of groups of stocks. The skeletons for the class definitions `PriceResponse` and `GroupedPriceResponse` are provided in appendix C.9 and C.10 respectively. The full class definitions are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

In addition to the classes, we used MATLAB's parallel computing toolbox (PCT) to expedite the data processing process by implementing data parallelism where possible.

In particular, for each market, we used *Parallel for loops* to simultaneously calculate the price response of different stocks. Refer to the MATLAB script in appendix C.11 for an illustration of how this was archived.

2.5 Concluding remarks

This chapter described the construction of the workflow that we used to analyse the price price response of trades for the stocks listed in appendix A. The next chapter presents the calculated price response results for each group of stocks and analysis the effect of liquidity (measured by average daily traded value and average daily volume) on price response.

Chapter 3

Empirical price response for emerging market stocks

This chapter discusses the empirical price response of trades for seven groups of stocks respectively listed on the Bombay Stock Exchange (BSE), Brasil Bolsa Balcão (B3), the Egyptian Exchange (EGX), the Johannesburg Stock Exchange (JSE), the Moscow Exchange (MOEX), the Nairobi Securities Exchange and the Shanghai Stock Exchange (SSE). For each considered stock groups, we start by providing the history and market structure of its corresponding stock exchange. We then do an exploratory data analysis for the data of the stock group, before considering empirical price response. Here, we seek to investigate the relationship between price response and liquidity, where liquidity is measured by average daily volume (ADV) and average daily traded value (ADTV).

3.1 Brasil Bolsa Balcão

Brasil Bolsa Balcão (B3) is one of the largest stock exchange in Brazil. Although it was only established in March 2017, the history of B3 dates back to 1820. Between the years from 1820 to 2017 several stock exchanges/financial markets providers emerged in Brazil and subsequently participated in mergers and/or acquisitions. B3 is a result of some of the mergers. The first of these institutions to emerge was *Bolsa de Valores do Rio de Janeiro* (BVRJ). BVRJ was inaugurated in 1820; this made it one of the oldest stock exchanges in Brazil. The second institution to emerged was *Bolsa Livre de São Paulo*. However, this initiative only lasted 14 months as it was unable to withstand the financial crises prevailing at that time. Nevertheless, the project resumed in 1895 as *Bolsa de Fundos Públicos de São Paulo*. This was then renamed to *Bolsa de Valores de São Paulo* (BOVESPA) in 1967 [46].

The third constituent of B3 emerged in 1917 after a group of business agents founded *Bolsa de Mercadorias de São Paulo* (BMSP). BMSP was established to protect buyers and sellers from fluctuations in agricultural price by organizing a forward market. The mergers and acquisitions that resulted in the formation of B3 began in 1991, six year after the formation of *Mercantile and Futures Exchange* (BM&F) — the forth constituent of B3 to emerge, established to consolidate and modernize derivative markets in Brazil. May 9, 1991 saw the merger of BMSP and BM&F, creating the *Brazilian Mercantile and Futures Exchange*. The title "BM&F" was maintained. Eleven years later, BVRJ was sold to BM&F [47]. This action was followed by a merger between BM&F and BOVESPA in May 2008 to form Brazil's largest stock exchange at the time, BM&F Bovespa [46, 48].

BM&F Bovespa then would go on to provide capital market access in Brazil for almost 9 years. However, in 2016, the board of directors of BM&F Bovespa and Cetip — a securities central clearing house established in 1984 — approved the merger of the operations of the two companions. As of March 2017, these two companies operate under the banner of *Brasil Bolsa Balcão*.

B3 equity market model

Currently, B3 operates an equity market trading segment with four different phases[1]:

1 **Order cancellation phase** (09:30 - 09:45) Permits the cancellation of orders from the previous trading session.

2 **Pre-opening auction** (09:45 - 10:00) Allows for the submission of orders for the calculation of the theoretical opening price.

3 **Continuous trading** (10:00 - 17:55) Continuous trading session.

4 **Cash market closing call** (16:55 - 18:00) Closing call for all cash market securities.

Equity market trading is conducted on a multi-asset electronic trading platform called *Puma Trading System* [49]. Participants can trade assets by using *orders* and/or *offers*. An order is defined as: " the action through which clients order a Brokerage Firm to buy or sell Assets, or the rights attached to them, on their behalf and under conditions specified by them" [50]. Thus, it is roughly equivalent to a market order. An offer is defined as: " the action through which a trader for a certain a Brokerage Firm registers its intention to buy or sell Assets or rights attached to them under the conditions specified by the customer" [50]. Thus, it roughly corresponds to a limit order. Furthermore, certain securities have designated market makers in the Puma Trading System [49, 51]. For a detailed outline of the equity market trading rules of B3, refer to [50] and [49]. For a list of some of the stock market markers, see [52].

Bovespa Index

This study consider's stocks that constitute the Bovespa Index. Appedix A.2 lists all of these stocks. The Bovespa Index is a total return index that is designed to track the average performance if Brazil's stock market [53]. The inclusion and exclusion criteria for the index can be found at [54]. Figure 3.1 illustrates the transaction prices and volumes of 30 constituents of the Bovespa index on the 4^{th} of November 2010, between the period from 11:00 to 17:45.

[1] Times are shown in GMT-03:00

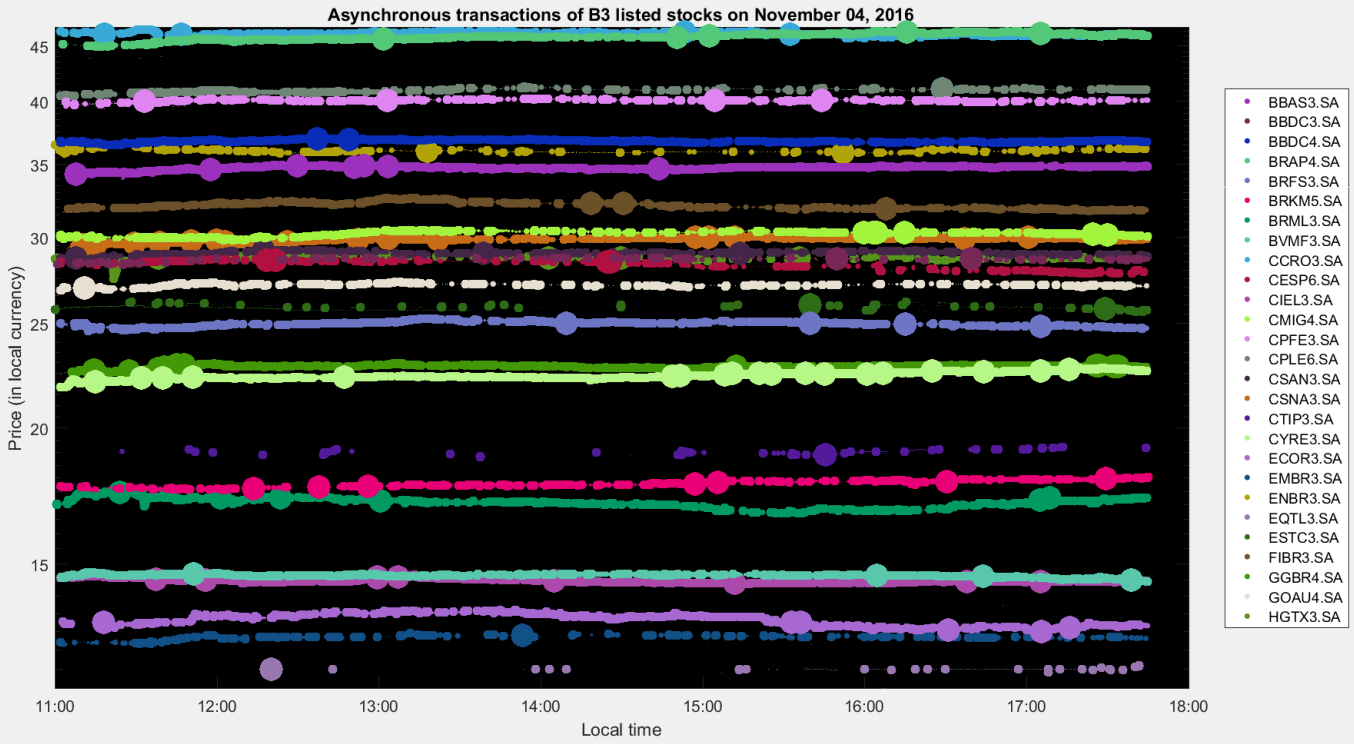


Figure 3.1: Transaction prices for B3 listed stocks on November 04, 2010. Each dot represents a transaction, and the size of the dot is proportional to the size of the transaction. The function provided in appendix C.7 was used to generate the scatter plot of each stock. The test script for this function is provided in appendix C.8.

Price impact of IBOVESPA constituents

Figure 3.2 visualises the price impact curves for constituents of the Bovespa Index. The figure on the left visualises the price impact curves for buyer initiated trades, and that on the right visualises the price impact curves for seller initiated trades, reflected about the horizontal axis. We note that the buyer and (reflected) seller price impact curves exhibit the same behaviour. That is, the measured price impacts tends to first decrease for small values of ω and then increase like a power-law for large ω.

In order to compare the price impacts of stocks with different liquidities, we created three liquidity bins: one for stocks with a relatively low liquidity, another for medium liquidity stocks and the last for high liquidity stocks. Stocks that fall under the low liquidity bin are plotted in red, those in the medium liquidity group are in green and those in the high liquidity group are in blue. In figure 3.2, average daily traded value (ADTV) was used as the proxy for proxy, and figure 3.3 uses average daily volume (ADV) as the proxy instead instead. For both proxies of liquidity, we note that stocks that have a high liquidity tend to have a lower magnitude of price impact, for all ω, for both buyer and seller initiated trades.

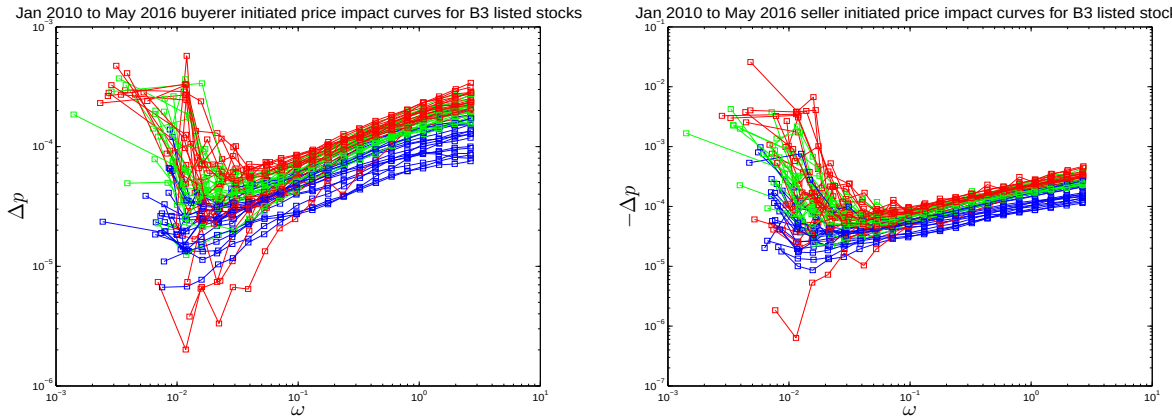


Figure 3.2: January 2010 to May 2016 price impact curves for constituents of the Bovespa index. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

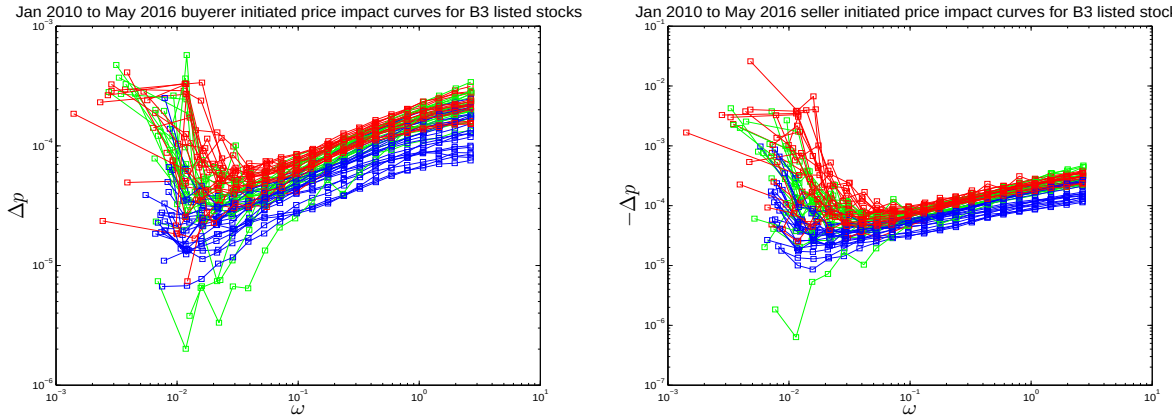


Figure 3.3: January 2010 to May 2016 price impact curves for constituents of the Bovespa index. Stocks were grouped into 3 different average daily volume (ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse

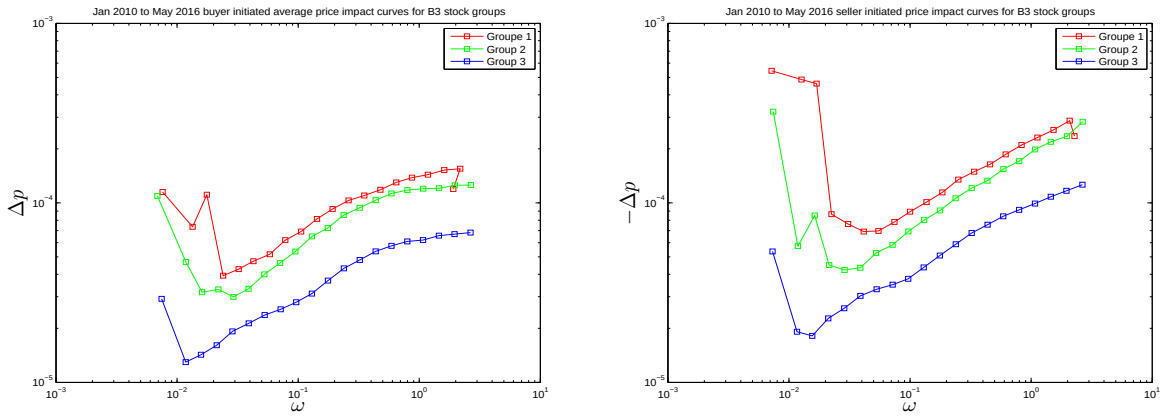


Figure 3.4: January 2010 to May 2016 average price impact curves for constituents of the Bovespa Index. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. The average price impact was then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	λ_b	α_b	λ_s	α_s
Group 1	0.4012×10^9	1.6975×10^7	8.1954×10^3	0.1672	5.9460×10^3	0.0191
Group 2	0.4110×10^9	2.6600×10^7	9.3909×10^3	0.2277	6.2073×10^3	0.2286
Group 3	1.3493×10^9	7.5033×10^7	1.6438×10^4	0.2868	1.0984×10^4	0.3182

Table 3.1: Calibrated price impact parameters for B3 stocks groups

In order to further investigate the aggregate behaviour and make quantitative comparisons between different stock groups, we used the method described in section 2.4.3 to consider the price impact curves of groups of stocks. Figure 3.4 illustrates the obtained results. We note that the price impact curves of the constructed stock groups striate themselves from top to bottom in increasing order of both ADVT and ADV.

We fit a linear regression of the form

$$\ln \Delta p = \alpha_j \ln \omega - \ln \lambda_j \tag{3.1}$$

to each measured price impact curves. The calibrated values of α_j and λ_j, with $j = \{b, s\}$, are shown in table 3.1. Here λ_b and α_b represent the calibrated parameters for the buyer initiated price impacts, and λ_s and α_s represent those for the (reflected) seller initiated price impacts. We note that as ADV and ADTV increase, both λ_j and α_j also increase.

Trade sign autocorrelation and price dispersion

Figure 3.5 visualises the calculated trade sign autocorrelation $\mathcal{C}(\ell)$ and normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the Bovespa Index. We note that all of the considered stocks have a $\mathcal{C}(\ell)$ that remains highly persistent, even after a large number of trades. We also note that all of the considered have a $\mathcal{D}(\ell)$ that exhibits sub-diffusion.

Using average daily volume as a proxy for liquidity, figure 3.6 indicates that stocks that have a high liquidity tend to have a smaller trade sign autocorrelation $\mathcal{C}(\ell)$ and smaller normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$ for all ℓ. As can be seen from figure 3.5, this relationship (between liquidity and $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$) is not as discernible when average daily traded value is used as the liquidity proxy.

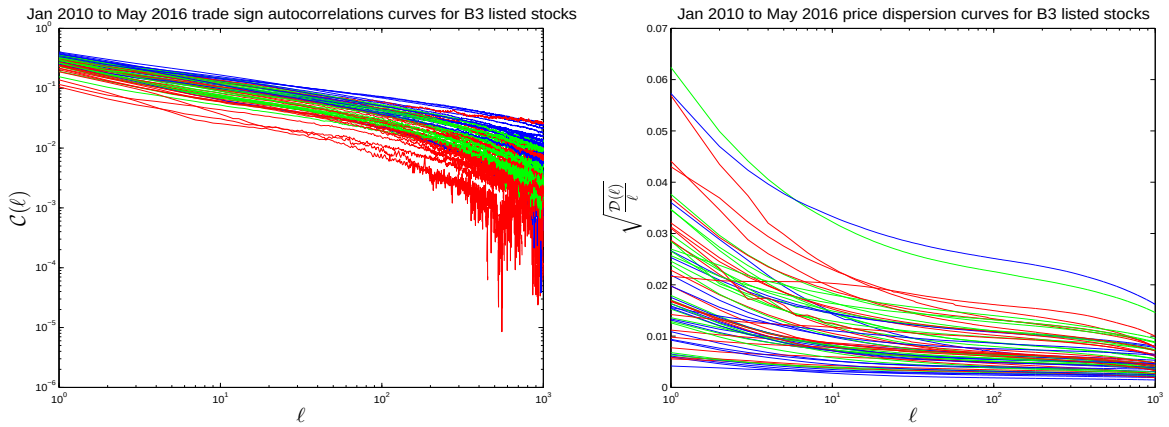


Figure 3.5: January 2010 to May 2016 daily average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and daily average normalised price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the Bovespa Index. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

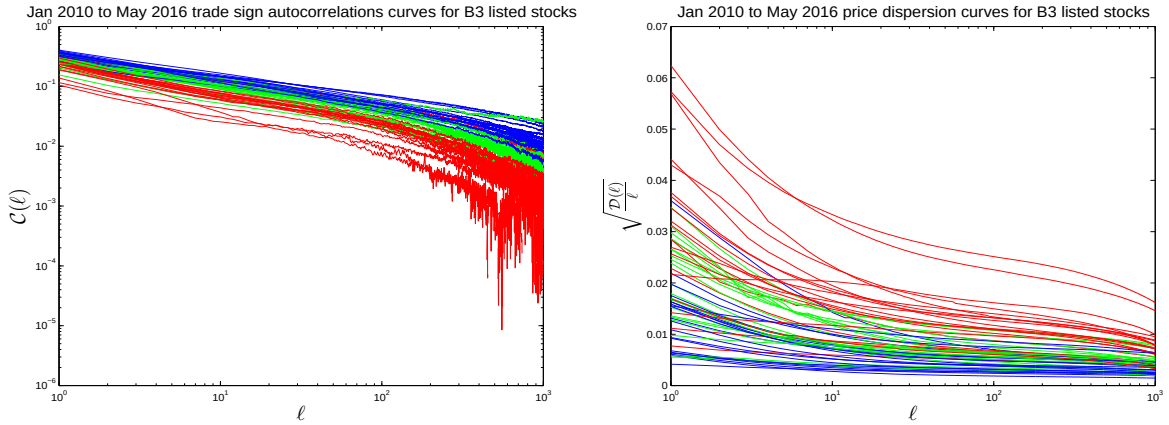


Figure 3.6: January 2010 to May 2016 daily average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and daily average normalised price dispersion curves $\sqrt{\mathcal{D}(\ell)/\ell}$ (right) for constituents of the Bovespa Index. Stocks were grouped into 3 different average daily volume (ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

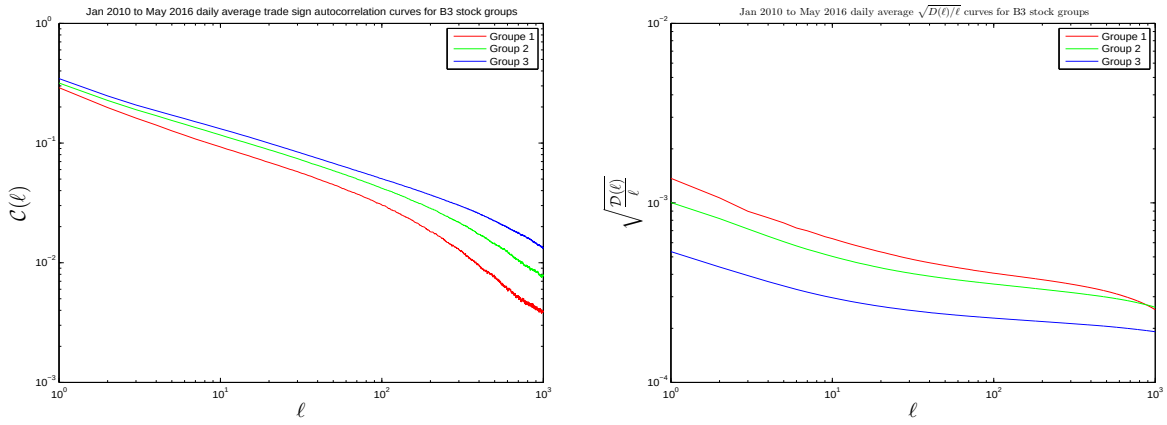


Figure 3.7: January 2010 to May 2016 daily average trade autocorrelation curves $\mathcal{C}(\ell)$ (left) and daily average normalised price dispersion curves $\sqrt{\mathcal{D}(\ell)/\ell}$ (right) for constituents of the Bovespa Index. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository `https://github.com/Telmakaza/Calculate_PriceResponse`.

Stock group	ADVT	ADV	$\mathcal{C}_0$	γ
Group 1	0.4012×10^9	1.6975×10^7	0.3144	0.5486
Group 2	0.4110×10^9	2.6600×10^7	0.3473	0.4884
Group 3	1.3493×10^9	7.5033×10^7	0.3588	0.4409

Table 3.2: Calibrated $\mathcal{C}(\ell)$ parameters for B3 stocks groups

Figure 3.7 visualises the daily average trade sign autocorrelation curves $\mathcal{C}(\ell)$ and daily average normalised price dispersion curves $\sqrt{\mathcal{D}(\ell)/\ell}$ for groups of stocks that constitute the Bovespa Index. We note for $\mathcal{C}(\ell)$, the stock groups are arrange, from bottom to top, in increasing order of ADV and ADTV. This means that stocks that have a high ADV and ADTV tend to have a high degree of trade sign autocorrelation. This observation is consistent with that made in figures 3.5 and 3.6.

After fitting a relationship of the form

$$\mathcal{C}(\ell) = \mathcal{C}_0 \ell^{\gamma} \tag{3.2}$$

to the $\mathcal{C}(\ell)$ of each stock group, we obtained the parameters indicated in table 3.2. We note that the value of $\mathcal{C}_0$ increases with ADV and ADTV. This is hardly surprising, since this is precisely the observation that we make in figure 3.5 and 3.6. We also note that the value of γ for each stock group is $0 < \gamma < 1$. According to Bouchoud *et al* (2008)[55], this may suggest that all of the considered stocks have a long memory trade sign autocorrelation process.

Turning our attention to $\sqrt{\mathcal{D}(\ell)/\ell}$, we note that even though the considered stocks may posses a long memory trade sign autocorrelation process, on average, according to the plot on the right in figure 3.7 all of the considered stock have a change price process that does exhibit long rage correlations. Furthermore, we note that stocks that high ADTV and ADV tend to experience a smaller value of $\sqrt{\mathcal{D}(\ell)/\ell}$ for all ℓ.

Price response coefficient and bare response

Figure 3.8 visualises the daily average price response coefficient $\mathcal{R}(\ell)$ and daily average bare response curves $G_0(\ell)$ for constituents of the Bovespa index. From the plot on the left in figure 3.8, we note that most of the considered stocks have a $\mathcal{R}(\ell)$ that initially increases fro small ℓ, and then decreases for values of ℓ. This finding is consistent with the observations made in [29] and [25]. Furthermore, from the plot on the left in figure 3.8, we note that all of the considered stocks have a $G_0(\ell)$ that decreases like a power low.

Turning our attention the figure 3.9, we note that when we use ADV a proxy for liquidity, liquid stocks tend to exhibit a smaller price response for all ℓ, where price response is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$. However, as can be seen in figure 3.8, this relationship is as not as discernible when ADTV is used as the liquidity proxy.

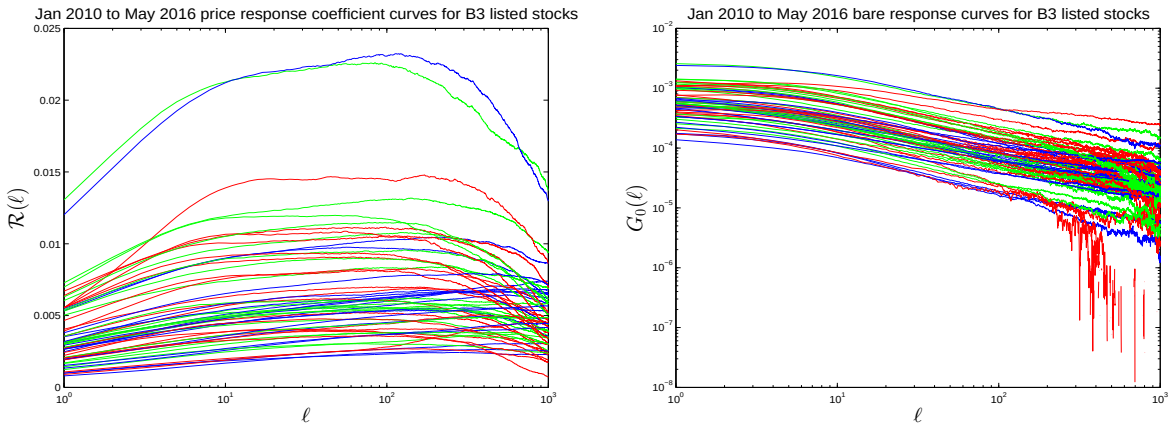


Figure 3.8: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the Bovespa Index. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

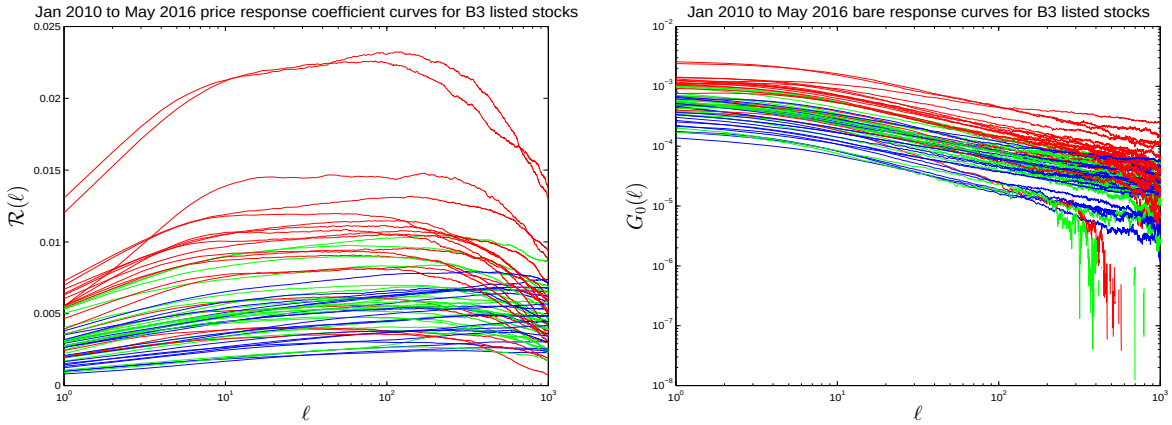


Figure 3.9: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the Bovespa Index. Stocks were grouped into 3 different average daily volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

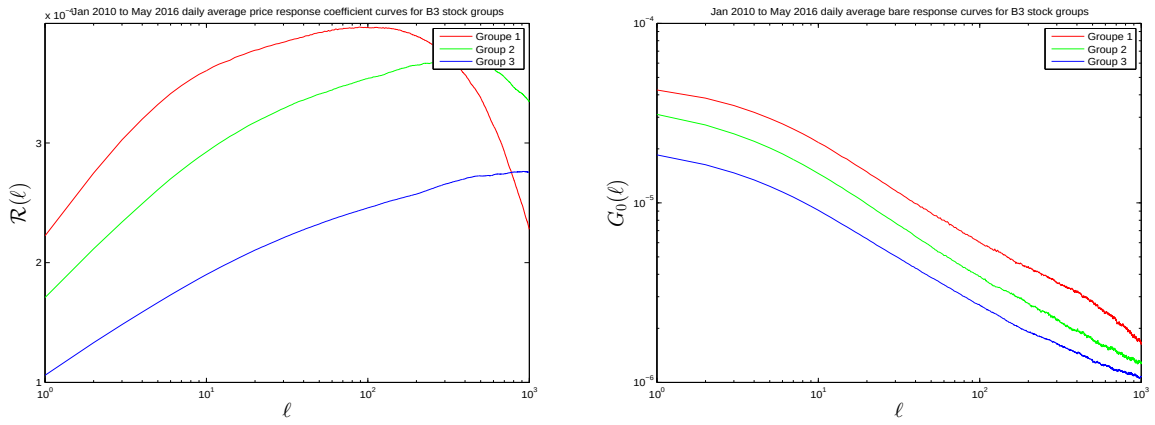


Figure 3.10: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ (left) and daily average bare response curves $G_0(\ell)$ (right) for constituents of the Bovespa Index. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{R}(\ell)$ and $G_0(\ell)$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	Γ_0	ℓ_0	β
Group 1	0.4012×10^9	1.6975×10^7	5.5235×10^{-5}	0	0.4754
Group 2	0.4110×10^9	2.6600×10^7	3.8795×10^{-5}	0	0.4909
Group 3	1.3493×10^9	7.5033×10^7	2.2591×10^{-5}	0	0.4519

Table 3.3: Calibrated $G_0(\ell)$ parameters for B3 stocks groups

Figure 3.10 visualises the measured $\mathcal{R}(\ell)$ and $G_0(\ell)$ for each constructed stock group. We note that stock groups that have a high ADV and ADTV tend to have a smaller magnitude of $\mathcal{R}(\ell)$ and $G_0(\ell)$ for most ℓ.

After fitting a relationship of the form

$$G_0(\ell) = \frac{\Gamma_0}{(\ell_0 + \ell)^{\beta/2}}, \tag{3.3}$$

to the measured $G_0(\ell)$, we obtained the parameters indicated in table 3.3. As suggested by figure 3.10, Γ_0 increases with ADV and ADTV. We also note that our calibrated value of β is comparable with that obtained by [25]. The latter obtained $\beta \in [0.12, 0.44]$. We find $\beta \in (0.45, 0.48)$. However, we note that we obtained an ℓ_0 value of 0 for all stock groups. [25] obtained $\ell_0 \in (0.2, 20.3]$.

Concluding remarks

This section considered the price response of stocks that constitute the Bovespa Index. We started by describing the stock exchange at which these stocks are listed (the B3) before providing a brief description of the Bovespa Index. We then considered the average price impact $\mathcal{R}(\ell = 1, V)$, the trade sign autocorrelation $\mathcal{C}(\ell)$, the normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$, the price response coefficient $\mathcal{R}(\ell)$ and the bare response function $G_0(\ell)$ for constituent stocks of the Bovespa Index. Using $\mathcal{R}(\ell = 1, V)$, $\mathcal{D}(\ell)$, $\mathcal{R}(\ell)$ and $G_0(\ell)$

respectively as measures of price response, we found that the higher the liquidity[2] of a stock, the smaller its price response. We found that this inverse relationship between price response and liquidity is more discernible when liquidity is measured in terms of ADV. This is illustrated by the contrast between figure 3.8 and 3.9. Lastly, we found that stocks that have a high ADV and ADTV tend to have a greater magnitude of trade sign autocorrelation $\mathcal{C}(\ell)$. This is illustrated by the plots on the left in figures 3.5 to 3.7 and by table 3.2.

We make an interesting observation in figures 3.2 and 3.3. Namely, we see that small trade volumes tend to have a large price impact. As suggested by Harvey *et al* (2017)[23], this may be caused by a lower level of *orderbook resilience* for small trade volumes. That is, the intensity of low volume trades are not followed by an equal intensity in quote replenishment. This results in a less resilient order book at lower trade volumes which in turn permits a higher price impact. We observe a similar behaviour in the South African market.

Another interesting observation that we make is found in figure 3.7. In this figure, there is a period following a trade where the autocorrelation decreases faster for less liquid stocks. We conjuncture that since less liquid stocks are more thinly traded, traders tend to execute a smaller number of child orders for these stocks. They may do this so as to minimize the effects of trading an illiquid stock.

The last observation that we make is that in figure 3.10, the less liquid stock tend to have a price response that is initially very high but then decays at a relatively fast rate. This implies that less liquid stocks in this market have a large temporary price impact. The more liquid stock, however, exhibit a more *transient* price impact.

[2]Here, liquidity was measured using average daily volume and average daily value traded

3.2 Bombay Stock Exchange

The Bombay Stock Exchange (BSE) is the largest stock exchange in India in terms of the number of listed companies and market capitalization. As of June 2017, it has more than 5000 listed companies and a market capitalization of close to $USD 2 trillion. The inception of the Bombay Stock Exchange can be tracked back to 1850's when a group of stock brokers gathered under banyan trees to conduct meetings. The group had to shift its trading venue several times because the number of brokers in the group was increasing and the brokers would overflow to the streets. In 1874 the brokers found a permanent venue in Dalal Street. In 1875 the BSE (then know as "The Native Share & Stock Brokers' Association") was established [56,57]. After operating an open out cry system of trading for 120 years, the BSE switched to an electronic trading platform called BSE On-Line Trading (BOLT) in 1995. 1997 saw the expansion of this system nationwide [58]. Today, the exchange claims to be the world's fasted exchange with a median response time of 6 microseconds [59,60].

BSE equity market model

Trading takes place on weekdays on the BSE, and each trading day is made up of four sessions:

1 **Pre-open** (09:00 to 09:15): Comprised of an order entry and an order matching period. The order entry period runs for 8 minutes [3] and allows for the entry, modification and cancellation of orders. Both limit and market orders are allowed in this period and there is dissemination of the indicative equilibrium price, indicative matchable quantity and indicative index values. The order matching period starts immediately after the order entry period concludes. Order additions, modifications and cancellations are not permitted in the order matching period since it is specifically reserved for opening price determination and trade conformation.

2 **Continuous trading** (09:15 to 15:30): In this session, orders are matched on a price-time priority and trades occur continuously. It commences only after the pre-opening session has concluded. Any unexcused orders in the pre-open period are moved to this session.

3 **Closing** (15:30 to 15:40): In this session, VWAP closing price is determined.

4 **Post close** (15:40 to 16:00): In this session, the execution of trades at the closing price is determined.

In edition to this, the BSE offers a separate block deal trading session to facilitate the execution of large trades. This session starts with the commencement of the continuous trading session. For more information regarding the market model of the BSE, refer to [61] and [62].

[3] The order entry period may not necessarily be exactly 8 minutes as there is system driven random stoppage between the 7^{th} and 8^{th}.

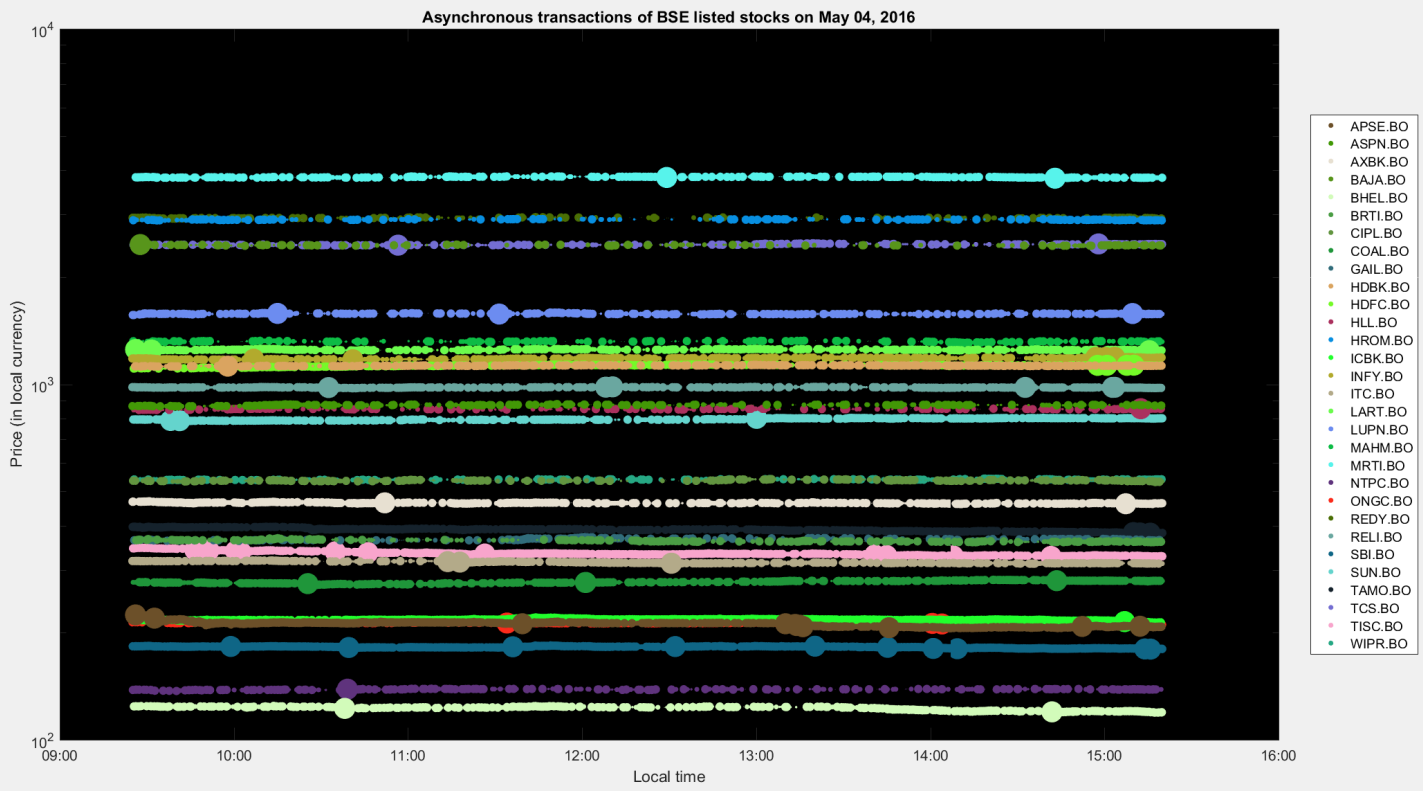


Figure 3.11: Transaction prices for BSE listed stocks on November 04, 2010. Each dot represents a transaction, and the size of the dot is proportional to the size of the transaction. The function provided in appendix C.7 was used to generate the scatter plot of each stock. The test script for this function is provided in appendix C.8.

BSE SENSEX index

Our investigation considered stocks that constitute the BSE SENSEX index — a free float market capitalization index. The value of this index (at any point in time) reflects the free float market value of 30 component stocks relative to some base period. Appendix A.1 list all of the component stocks. Figure 3.11 illustrates the transaction prices and volumes for constituents of the BSE SENSEX index on the 4^{th} of May 2016. Below, we consider the empirical price response for constituents of this index for the period from January 2010 to May 2016.

Price impact of BSE SENSEX constituents

Figure 3.12 and 3.13 visualises the average price impact functions of BSE SENSEX constituents for the period from January 2010 to May 2016. We note that all of the considered stocks have price impact functions that increase for all values of ω. Furthermore, we note that the behaviour of buyer initiated price impact curves is similar to that of the seller initiated, reflected about the horizontal axis.

Using average daily value traded as a proxy for liquidity, from figure 3.12, we note that stocks that have a high liquidity tend experience a smaller magnitude of price impact for all ω. As can be seen in figure 3.13 this relationship (between liquidity and price impact) is also observed when average daily volume is used as the liquidity proxy.

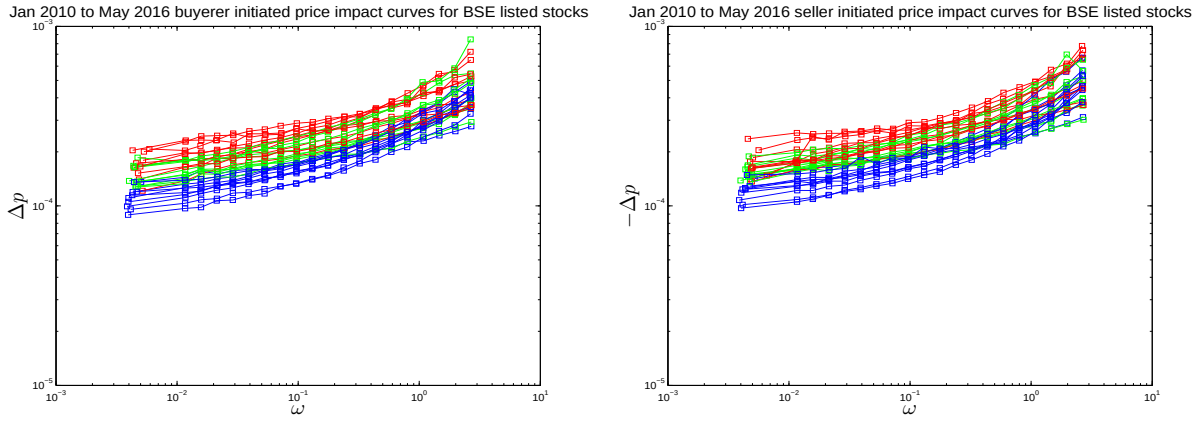


Figure 3.12: January 2010 to May 2016 price impact curves for constituents of the BSE SENSEX. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

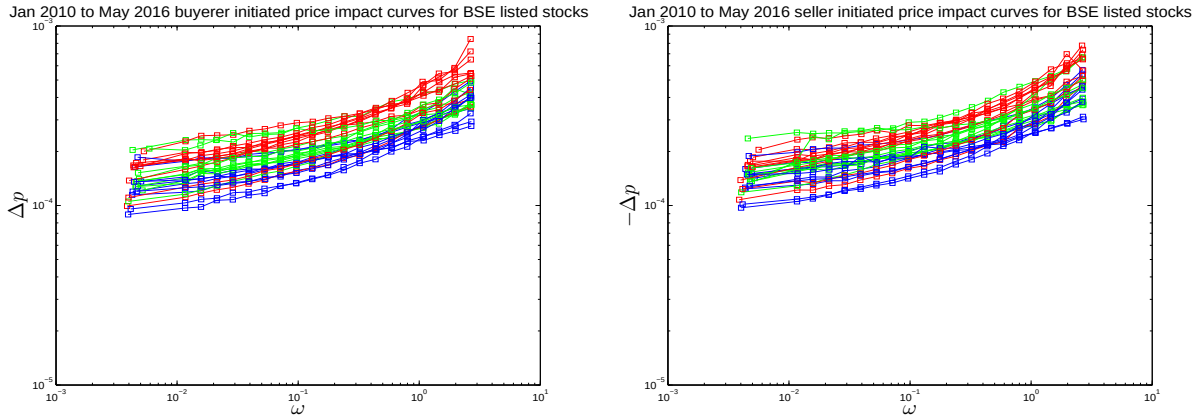


Figure 3.13: January 2010 to May 2016 price impact curves for constituents of the BSE SENSEX. Stocks were grouped into 3 different average daily volume (ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse..

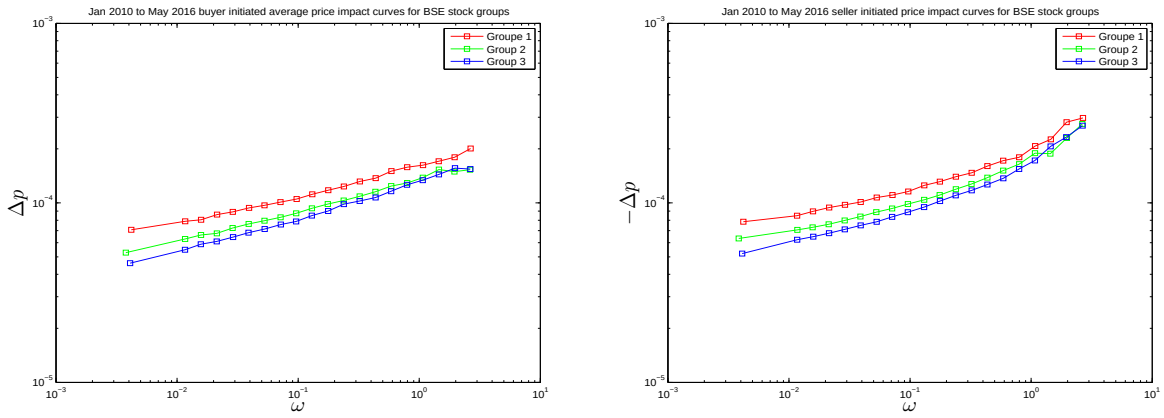


Figure 3.14: January 2010 to May 2016 average price impact curves for constituents of the BSE SENSEX. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. The average price impact was then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	α_b	λ_b	α_s	λ_s
Group 1	7.9492×10^8	1.3664×10^6	0.1634	6.2433×10^3	0.2057	4.9385×10^3
Group 2	9.5174×10^8	1.5694×10^6	0.1724	7.4710×10^3	0.2197	5.6609×10^3
Group 3	1.7878×10^9	2.8197×10^6	0.1960	7.7118×10^3	0.2457	5.8723×10^3

Table 3.4: Calibrated price impact parameters for BSE SENSEX stock groups.

In order to make more quantitative comparisons between different stock groups, we used the stock grouping method described in section 2.4.3 to create three stock groups labelled Group 1 to 3 respectively. Table 3.4 indicates the ADTV and ADV of each stock group. From figure 3.14 and table 3.4 we note that as ADTV and ADV increases, the magnitude of price impact decreases.

After fitting a relationship of the form

$$\ln \Delta p = \alpha_j \ln \omega - \ln \lambda_j \tag{3.4}$$

to the measured price impacts, we obtained the parameters indicated in table 3.4. Here, α_s and β_s are the calibrated parameters for the buyer initiated price impacts, and α_s and λ_s are those for the seller initiated price impacts. As suggested by by figure 3.12, 3.13 and 3.14, we note that as ADTV and ADV increase, λ_i (where $j = \{s, b\}$) decreases — corresponding to a downward shift in the price impact curve. We also note that α_j increases with ADTV and ADV. This implies that stocks with a relatively high ADTV and ADV tend to have a relatively steep price impact.

Trade sign autocorrelation and price dispersion

Figure 3.15 visualises the daily average trade sign autocorrelation curves $\sqrt{\mathcal{D}(\ell)/\ell}$ and daily average normalised price dispersion curves for constituents of the BSE SENSEX. From the left plot in figure 3.16, we note that all of the considered stocks have a trade sign autocorrelation process that remains persistent, even after a large number of trades. In addition to this, from right plot in figure 3.15, we note that each stock has a $\sqrt{\mathcal{D}(\ell)/\ell}$ that exhibits sub-diffusive properties, implying that price changes are not correlated.

Taking ADV as a proxy for liquidity, as indicated by the blue plots in figure 3.15, we note that the more liquid stocks tend to have a larger degree of trade sign autocorrelation $\mathcal{C}(\ell)$ and a smaller normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$. We also note that if we use ADTV as a proxy for liquidity, as can be seen from the plot on the right in figure 3.15, the inverse relationship between liquidity and $\sqrt{\mathcal{D}(\ell)/\ell}$ is not as discernible as when ADV is used as the proxy.

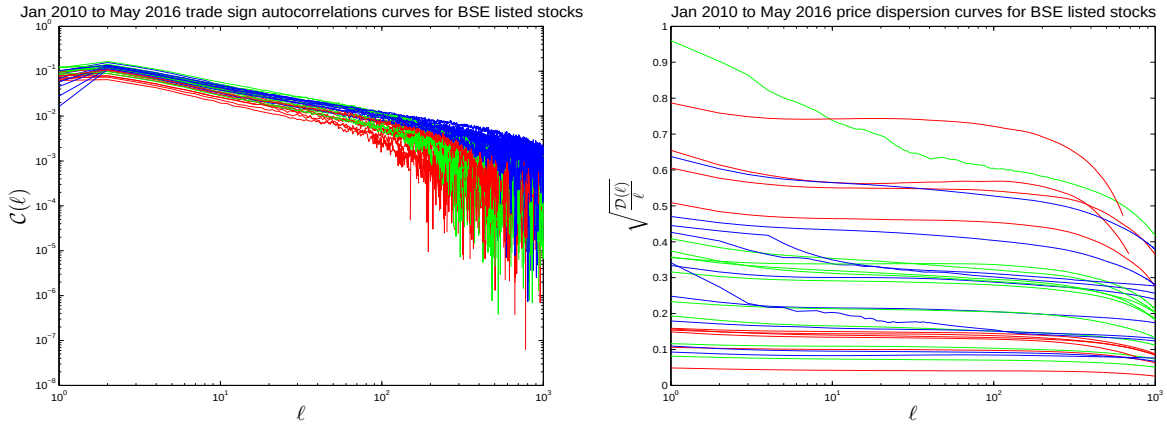


Figure 3.15: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the BSE SENSEX. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

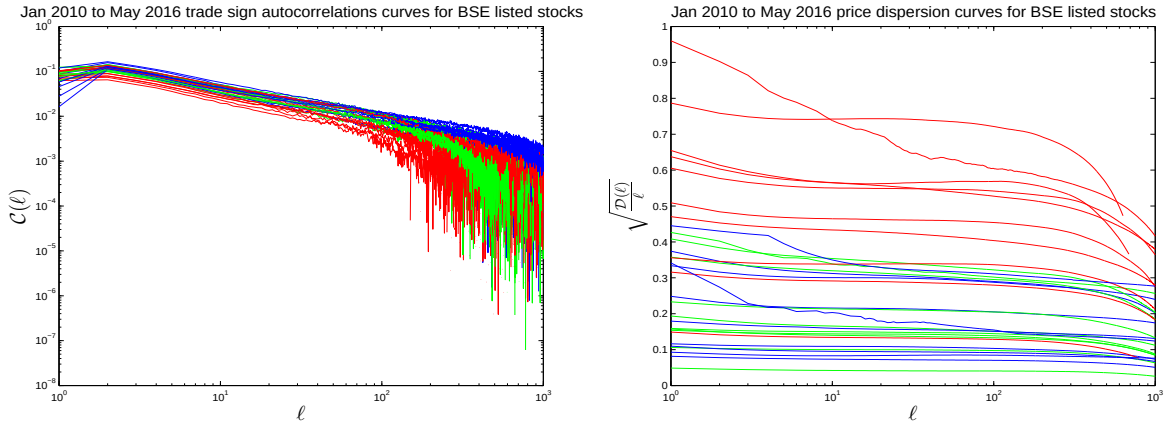


Figure 3.16: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the BSE SENSEX. Stocks were grouped into 3 different average daily volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

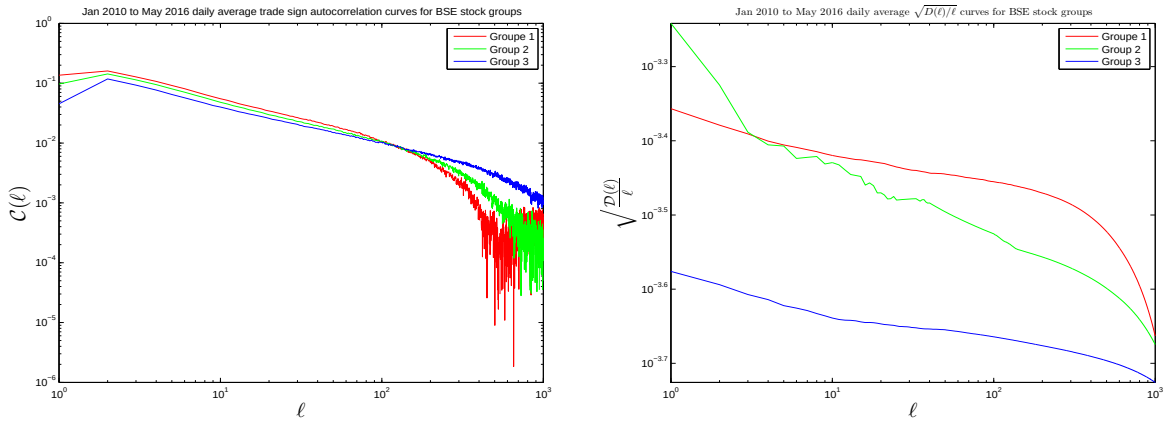


Figure 3.17: January 2010 to May 2016 daily average trade autocorrelation curves $\mathcal{C}(\ell)$ (left) and daily average normalised price dispersion curves $\sqrt{\mathcal{D}(\ell)/\ell}$ (right) for constituents of the BSE SENSEX. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	$\mathcal{C}_0$	γ
Group 1	7.9492×10^8	1.3664×10^6	0.2049	0.6611
Group 2	9.5174×10^8	1.5694×10^6	0.1672	0.6278
Group 3	1.7878×10^9	2.8197×10^6	0.1165	0.5444

Table 3.5: Calibrated $\mathcal{C}(\ell)$ parameters for BSE stocks groups

After grouping, stocks similar to the way we did for price impact, we calculated $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ of each constructed stock group and obtained the results shown in figure 3.17. Contradictory to our observation in figure 3.16, the left plot in figure 3.17 and table 3.5 show that stocks that have a low ADV and ADTV tend to have a higher degree of trade sign correlation for small values of ℓ. However, as ℓ increases, the more actively traded stocks tend to have a higher value of $\mathcal{C}(\ell)$.

After fitting the measured $\mathcal{C}(\ell)$ to a relationship of the form

$$\mathcal{C}(\ell) = \mathcal{C}_0 \ell^{\gamma} \tag{3.5}$$

we obtained the parameters listed in table 3.5. We note that as ADV and ADTV increase, $\mathcal{C}_0$ decreases, corresponding to a vertical shift in $\mathcal{C}(\ell)$. Furthermore, we note that for all of the stock groups, $0 < \gamma < 1$, suggesting that the trade sign autocorrelation process of each stock group might be a memory process. However, the plot on the right figure 3.17 show that the price changes for all the stock groups are not correlated. In the framework of [29], this suggest that the impact of a single trade for all of these stock groups decays like a power law.

In addition to this, we note that the $\sqrt{\mathcal{D}(\ell)/\ell}$ of the constructed stock groups are arranged from top to bottom in increasing order of ADV and ADTV. This implies that the more actively trades stock exhibit a smaller degree of price dispersion.

Price response coefficient and bare response

Figure 3.18 visualises the daily average price response coefficient curves $\mathcal{R}(\ell)$ and daily average bare response curves $G_0(\ell)$ for constituents of the BSE SENSEX. We note that all of the considered stocks have a $\mathcal{R}(\ell)$ that first initially increases, and then decreases for large ℓ. We also note that most of the considered stocks experience a daily average bare response $G_0(\ell)$ that decreases like a power law.

Using ADV as a proxy for liquidity, figure 3.19 shows that more liquid stocks tend to have a smaller price response, where price response if measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$. However, as can be seen from figure 3.18, this relationship (between liquidity and price response) is arguably less discernible when price ADTV is used as the liquidity proxy.

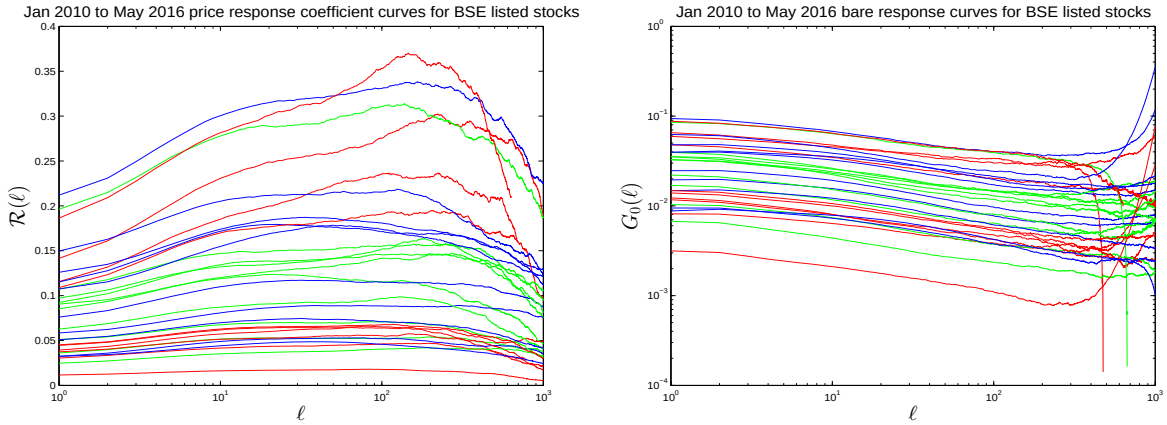


Figure 3.18: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the BSE SENSEX. Stocks were grouped into 3 different average daily traded value(ADTV) groups. Plots in red represent stocks in the low ADTSV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

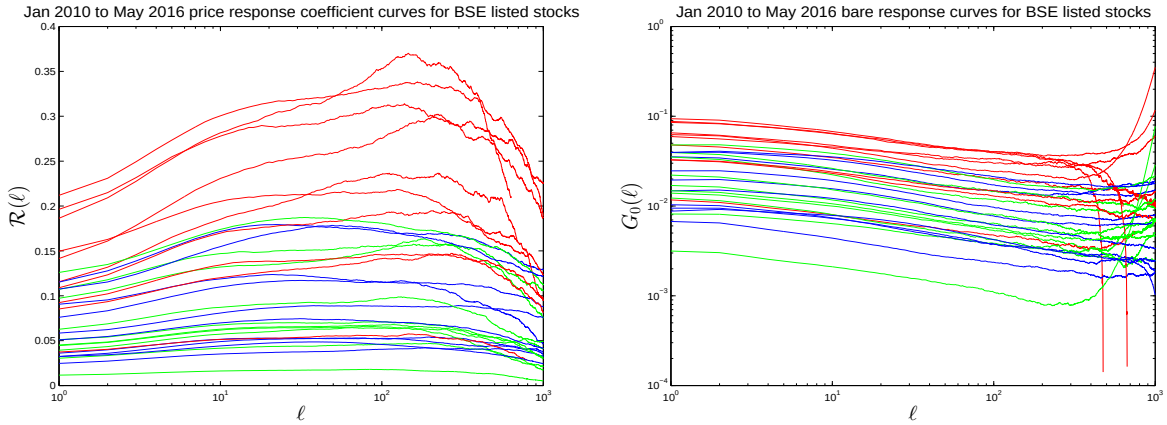


Figure 3.19: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the BSE SENSEX index. Stocks were grouped into 3 different average daily traded volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse

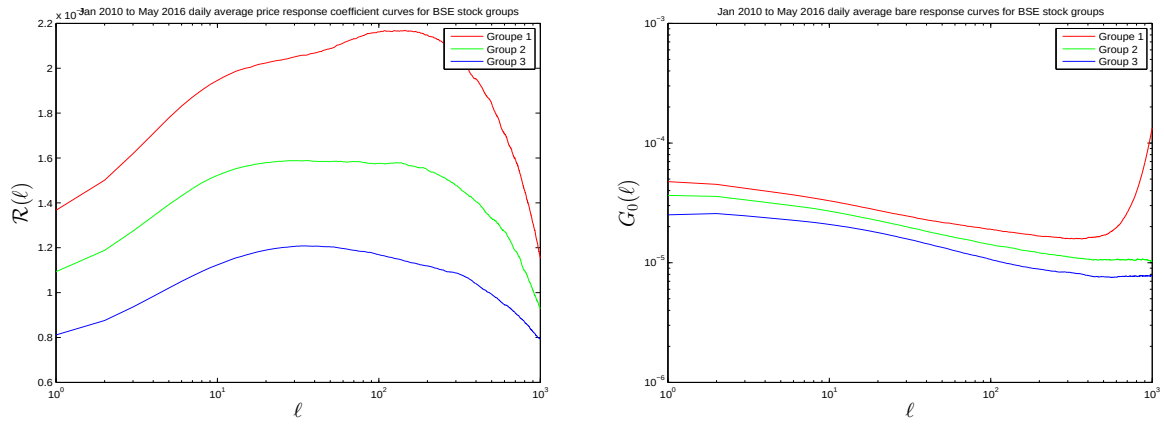


Figure 3.20: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ (left) and daily average bare response curves $G_0(\ell)$ (right) for constituents of the BSE SENSEX. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{R}(\ell)$ and $G_0(\ell)$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	Γ_0	ℓ_0	β
Group 1	7.9492×10^8	1.3664×10^6	1.1538×10^{-5}	0	-0.8959
Group 2	9.5174×10^8	1.5694×10^6	3.8681×10^{-5}	0	0.2051
Group 3	1.7878×10^9	2.8197×10^6	3.0424×10^{-6}	0	0.2166

Table 3.6: Calibrated $G_0(\ell)$ parameters for BSE stocks groups

Figure 3.20. Visualises the January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ (left) and daily average bare response curves $G_0(\ell)$ for constituents of the BSE SENSEX. From table 3.6 and figure 3.20 we note that stocks that have a relatively high ADV and ADTV tend to experience a smaller price response, where price response if measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$.

Concluding remarks

This section considered the price response of stocks that constitute BSE SENSEX index. We started by describing the stock exchange at which these stocks are listed (i.e the Bombay Stock Exchange) and then we provided a brief overview of the BSE SENSEX index. After this, we considered the price response of stocks that constitute this index. We found that the more actively traded stocks (those with a relatively high ADTV and ADV) tend to experience a smaller price response, where price response is measured by price impact $\mathcal{R}(\ell = 1, V)$, normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$, the price response coefficient $\mathcal{R}(\ell)$ and bare response $G_0(\ell)$. Furthermore, we found that less actively traded stocks tend to have a high degree of trade sign autocorrelation $\mathcal{C}(\ell)$ for most ℓ.

Another observation that we made for this market is that the price impact "hump" that we saw for small trade volumes in B3 listed stocks (figure 3.4) is not found in the stocks that we considered here (figure 3.14). We conjecture that the order book resilience in this market is more consistent across all trading volumes.

3.3 Egyptian Exchange

The Egyptian exchange (EGX) is one of the oldest stock markets in the middle east. It comprises the the Alexandria Stock Exchange and the Cairo Stock Exchange. Thus, its origins can be traced back to the establishment of these two exchanges. The first to emerge was the Alexandria Exchange (sometimes referred to as *Alexandria's Future Markets*)[63]. Alexandria's Future Markets was one of the oldest in the world in the 19^{th} century. It was established in 1883 and operated a market in cotton and a variety of cotton seeds. The first locally recorded cotton transaction took place in 1885. Cotton deal makers would wait for news-sheets from Europe which they used to guide them in future operations. As business grew, cotton deal markers moved to a nearby building and the *Association Cottoniere d'Alexandrie* (later renamed to *Alexandria General Produce Association* or AGPA) was established. This initiative offered trading services in cotton, cotton seeds and cereal both in the spot and future market. In 1899, the AGPA moved to a new building. The Exchange became a city landmark that featured in postcards, books and even city guides [63].

Meanwhile (back in Cairo), during an informal meeting in a café, Cairo's merchants and brokers were reminded by their leader, Monsieur Maurice Cattaui, that they should follow suit in Alexandria's footstep and establish their own stock exchange. Finally, in 1903, (14 years after the AGPA decided to a move a new building) an ad hoc committee presided by Monsieur Maurice Cattaui established the Cairo Stock Exchange. The committee temporarily leased the premises of Ottoman Bank (today Groppi-Adly Branch) while they initiated an international competition for the design of an exchange at Cairo's European district. The competition was settle in April 1907 and the prize went to a French architect, Raoul Brandon. However, the construction of the building did not proceed as a financial crisis (termed *The Panic of 1907*) prevailed later that year. However, 18 months after Brandon was awarded the prize, the *Corporation of Agents de Change* commissioned the Cairo firm of Edward Matasek and Maurice J. Cattaui to design and build an Exchange building. In 1928, the Cairo Exchange moved to its current trading venue on Sherifein Street. Incidentally, this was also the first meeting place for Cairo's speculative traders prior to the establishment of the Cairo Exchange. Before July 1961, the Cairo Exchange and Alexandria Exchange merged and the resulting exchange was listed fourth in the world. Although the two stock exchanges were merged, their trading activities were separate. It was only in 1996, after both bourses migrated to electronic trading, that the two unified trading [63]. Currently, both exchanges constitute the EGX and by the presidential decree of the year 1997, the EGX has two locations: Cairo and Alexandria. Nevertheless, it has one chairman and one board of directors [64].

Upon migrating to electronic trading, the EGX used the locally developed trading system for almost 9 years — after which it contracted with a Canadian software company to procure a new trading, clearing and settlement system, EFA Horizon. May 2001 saw the institution of the trading component of this system to the EGX [65]. In an effort to keep with the latest advancement in technology, the EGX replaced the EFA Horizon trading system with its current system, *X-Stream*. X-Stream is powered by NASDAQ OMX's X-Stream trading technology. It was designed to support increasing trading volumes of the EGX. Further, it is a multi-asset platform capable of handling equities, debt instruments, commodities, futures and options [65, 66].

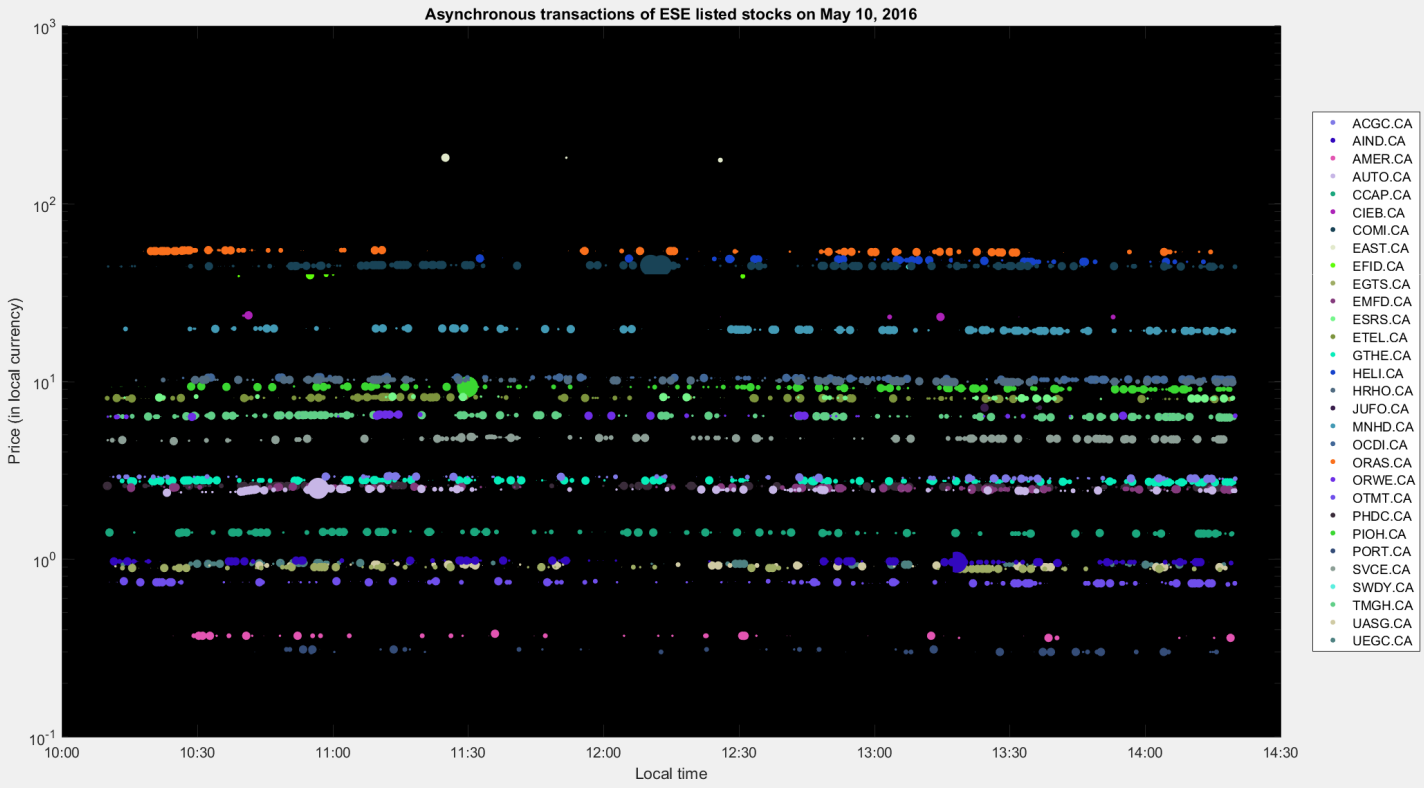


Figure 3.21: Transaction prices for EGX listed stocks on May 10, 2016. Each dot represents a transaction, and the size of the dot is proportional to the size of the transaction. The function provided in appendix C.7 was used to generate the scatter plot of each stock. The test script for this function is provided in appendix C.8.

EGX equity market model

Each trading day on the EGX is made up of two sessions: A *Discovery Session* and an *Official Trading Session*[67]. Through brokers, investors can trade directly with each other on the exchange. Thus, it is predominately an order driven market. All trading is carried out electronically, and orders are matched on price-time priority. For a detailed description of the EGX market structure, refer to [67] and [68].

EGX30 index

This investigation considers stocks that constitute the EGX30 — a market capitalization weighted index on the EGX. The index is calculated in a local currency and it has a start date of January 2, 1998 with a base value of 1000 points. Further, it is made up of the 30 most liquid stocks on the EGX with free float of at least 15%. Appendix A.3 lists the constituent stocks of this index. Figure 3.21 illustrates the transactions prices and volumes of EGX30 constituents on the 10^{th} of May 2016. Below, we consider the price response of trades for these stocks.

Price impact of EGX30 constituents

Figure 3.22 and 3.23 visualise the January 2010 to May 2016 measured price impact for constituents of the EGX30. We note that the measured price impacts do not exhibit the behaviour that we observed with B3 and BSE considered stocks. In particular, the price impacts do not increase like a power law, even for large values of ω. Moreover, the price impacts of the more actively traded stocks (indicated by blue lines in figures 3.22 and 3.23) are relatively flat for large ω.

Using ADTV as a proxy for liquidity, figure 3.22 shows that more liquid stocks tend to experience a lower degree of price impact. As can be seen in figure 3.23, this inverse relationship (between liquidity and price impact) is also observed when ADV is used as the liquidity proxy.

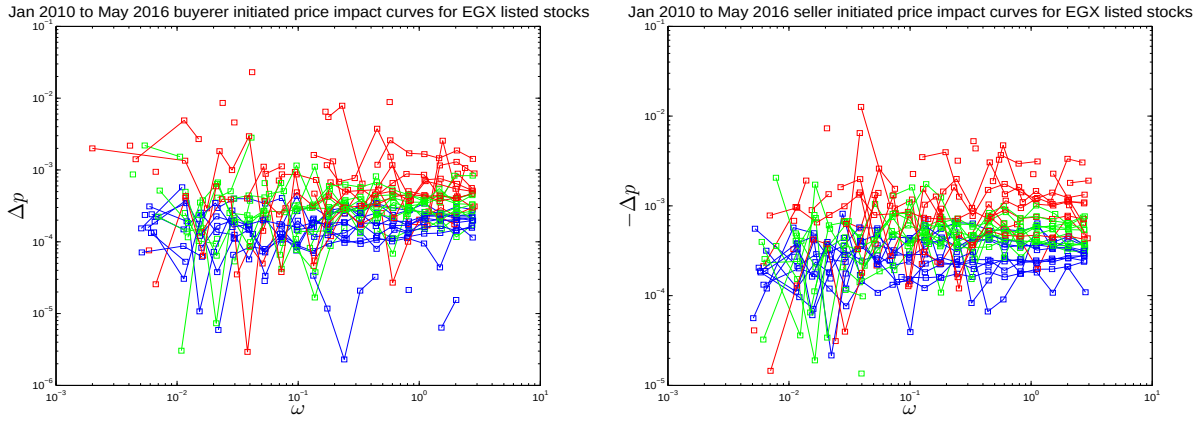


Figure 3.22: January 2010 to May 2016 price impact curves for constituents of the EGX30. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

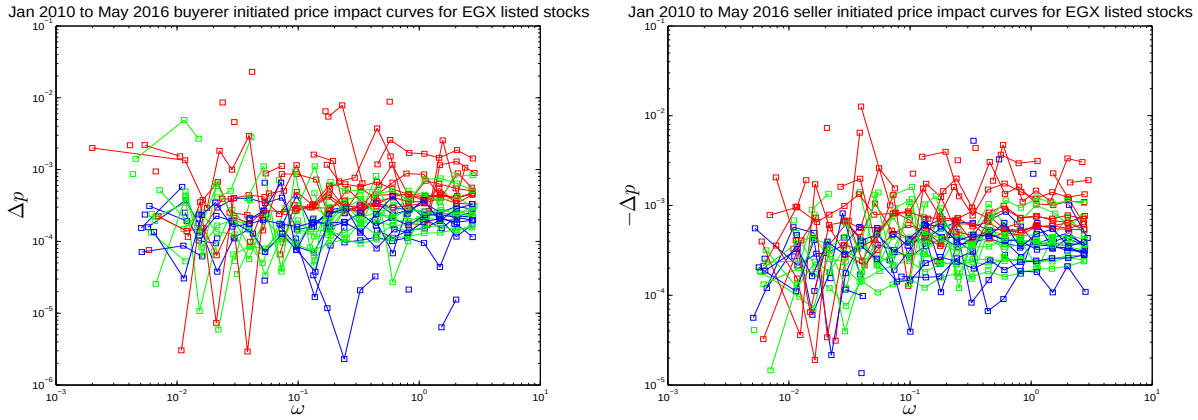


Figure 3.23: January 2010 to May 2016 price impact curves for constituents of the EGX30. Stocks were grouped into 3 different average daily volume (ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/.

Trade sign autocorrelation and price dispersion

Figure 3.24 visualises the daily average trade sign autocorrelation curves $\mathcal{C}(\ell)$ and daily average normalised price dispersion curves $\sqrt{\mathcal{D}(\ell)(\ell)/\ell}$ of constituents of the EGX30. We note that all of the considered stocks have a trade sign autocorrelation process $\mathcal{C}(\ell)$ that is not as persistent as that observed B3 and BSE listed stocks. We also note that the considered stocks have a $\mathcal{D}(\ell)$ that exhibits sub-diffusive properties, implying that price changes are not correlated.

Turning our attention to figure 3.25, we note that when we use ADV as a proxy for liquidity, the more liquid stock tend to experience a larger trade sign autocorrelation $\mathcal{C}(\ell)$ and smaller normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$. This relationship between liquidity and $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell/\ell)}$ becomes less discernible when ADTV is used as the liquidity proxy.

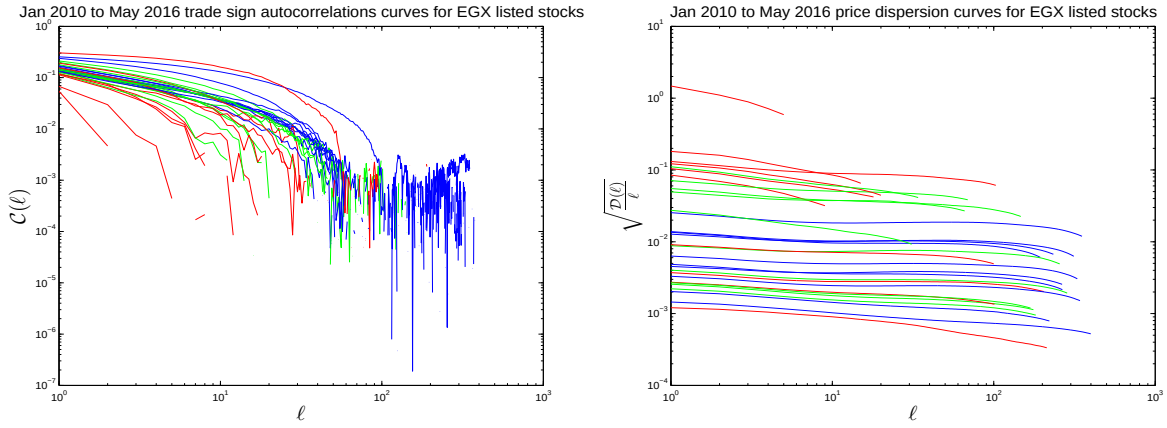


Figure 3.24: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the EGX30. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

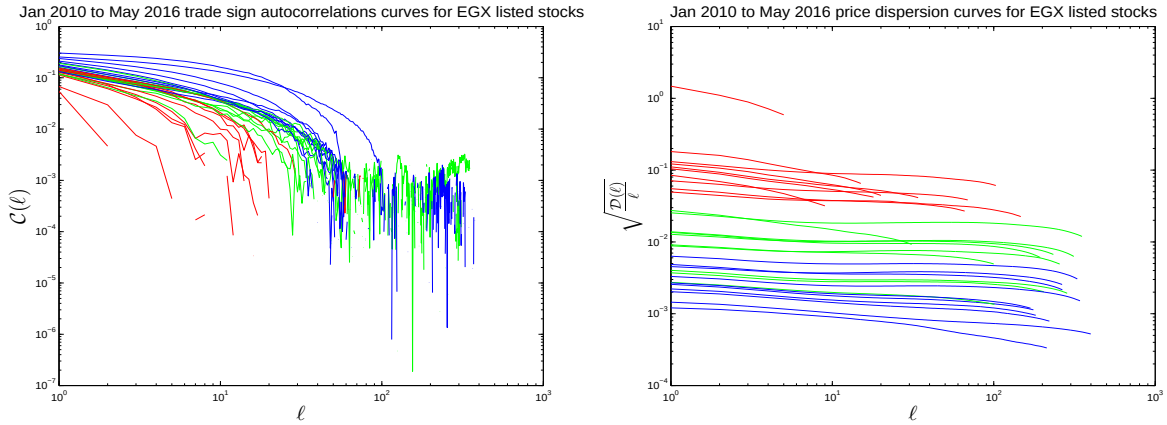


Figure 3.25: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the EGX30. Stocks were grouped into 3 different average daily volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/.

Price response coefficient and bare response

Figure 3.26 visualises the January 2010 to May 2016 daily average price response coefficient curve $\mathcal{R}(\ell)$ and bare response function $G_0(\ell)$ for constituents of the EGX30. We note that all of the considered stocks have a $\mathcal{R}(\ell)$ that increase for small values of ℓ, and then decreases for large ℓ. We also note that every considered stock has $G_0(\ell)$ that decreases for all ℓ.

Turning our attention to figure 3.27 we note that if we use ADV as a proxy for liquidity, stocks that have a high liquidity tend to have a small price response, where price response is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$. Further, figure 3.26 shows that the inverse relationship between liquidity and price response is less apparent when ADTV is used as the liquidity proxy.

Summary of results

This section considered the price response of stocks that constitute the EGX30 index. We started by describing the listing exchange for these stocks, The Egyptian Exchange, and then we gave a brief overview of the EGX30 index. We then calculated January 2010 to May 2016 average price impact $\mathcal{R}(\ell = 1, V)$, daily average trade sign autocorrelation $\mathcal{C}(\ell)$, daily average normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$, daily average price response coefficient $\mathcal{R}(\ell)$ and daily average bare response $G_0(\ell)$. We found that stocks that have a relatively high ADV tend to have a small price response, where price response is measured by $\mathcal{R}(\ell = 1, V)$, $\sqrt{\mathcal{D}(\ell)/\ell}$, $\mathcal{R}(\ell)$ and $G_0(\ell)$. Furthermore, we found that stocks that have a high ADV tend to exhibit a greater degree of trade sign autocorrelation $\mathcal{C}(\ell)$.

Turning our attention to the relationship between ADTV and price response, we found that stocks that have a relatively high ADTV tend to a smaller price impact $\mathcal{R}(\ell = 1, V)$. However, we did not observe a clear relationship between ADTV and $\mathcal{R}(\ell)$, $\mathcal{C}(\ell)$, $\sqrt{\mathcal{D}(\ell)/\ell}$ and $G_0(\ell)$.

However, given the sparse liquidity in this market, it is difficult to assert these observations. From figure 3.22 and 3.23, we note that the price impact curves for some of the illiquid stocks are broken lines. This is an artefact of the plotting system that we used in MATLAB: price impacts for some volumes are negative and these are not included in the log-log plot. Negative price impact are not what we expected. We conjuncture that the illiquidity of the stocks permitted this phenomena. The same can be said for NSE listed stocks (which we consider in section 3.6).

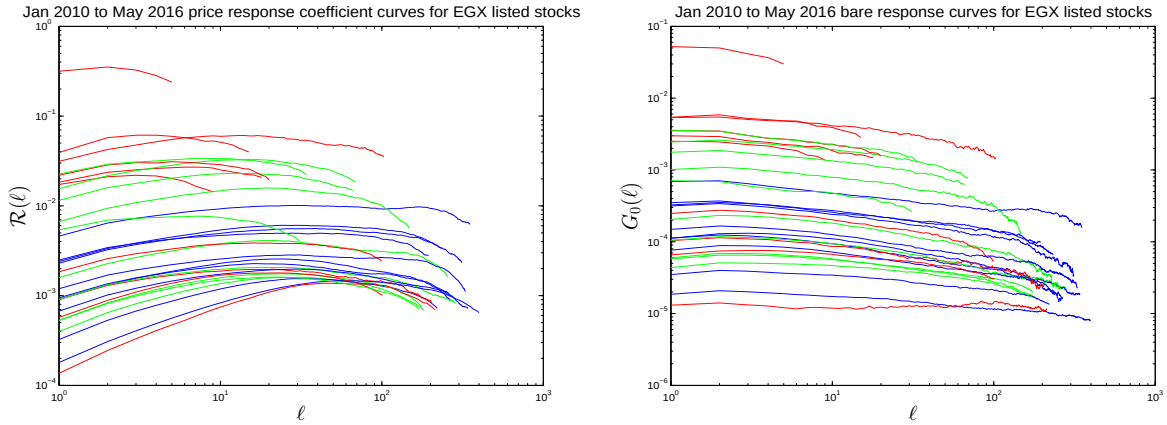


Figure 3.26: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the EGX30. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

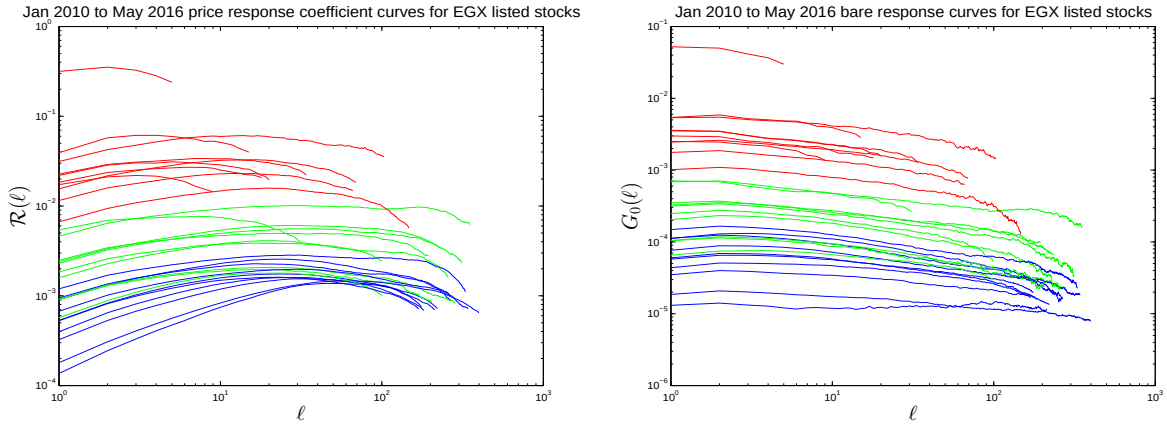


Figure 3.27: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the EGX30. Stocks were grouped into 3 different average daily traded volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

3.4 Johannesburg Stock Exchange

The Johannesburg Stock Exchange (JSE) is the largest stock exchange in African both by market capitalization and the number of listed companies[69]. As of June 2017, its market cap stands just above US$1 trillion and it has a total of 375 companies listed — of which 303 are foreign owned and 72 are domestically owned [69]. The JSE is not the first stock exchange in South Africa, however, it is currently the oldest that is still in operation [70]. It was established by Benjamin Wollan in 1887 after the discovery of gold (in Langlaagte) by George Harrison on July 1886 [71]. November 8, 1887 saw the official opening of the institution. The construction of the trading venue was still in progress when the exchange opened, therefore, tent-styled informal trading was conducted — with transactions taking place at bar counters, make-shift offices and in the open air [72]. However, by the end of the year, the construction of the trading venue was complete and it would become the preferred trading facility.

In 1963, the JSE became a member of the World Federation of Exchanges, and June 7, 1996 saw the ousting of the open out cry system of trading from the JSE. The JSE has since adapted an electronic system of trading and it has changed its trading platform several times across the time period from 1996 to 2012 [73–75].

JSE Equity market model

Currently, the JSE operates an order driven central orderbook with an opening auction, a closing auction, and a continuous trading period. There are 4 different order types on the JSE: limit orders, market orders, stop orders and stop limit orders. Further, the exchange is made up of a total of 7 trading segments, where each segment comprises a logical grouping of instruments and has a segment. For this study, we consider stocks in the *Top Companies* trading segment, with segment code ZA01. Most of these stocks are the constituents of the JSE Top40 index (JTOPI) and have the highest liquidity rating on the JSE. For a more detailed consideration of the market structure of JSE, refer to [45].

JSE Top 40 index

For the JSE portion of this study, we only consider stocks that comprise the JSE Top40 Index. The JSE Top40 index is a capitalization weighted index that comprises the 40 largest stocks (by market cap) listed on the JSE [76, 77]. Appendix A.4 lists all of the JSE Top40 constituents from January 2010 to May 2016. Figure 3.28 illustrates the transactions prices and volumes of JSE Top40 constituents on the 4^{th} of May 2016. Below, we consider the price response of trades for these stocks.

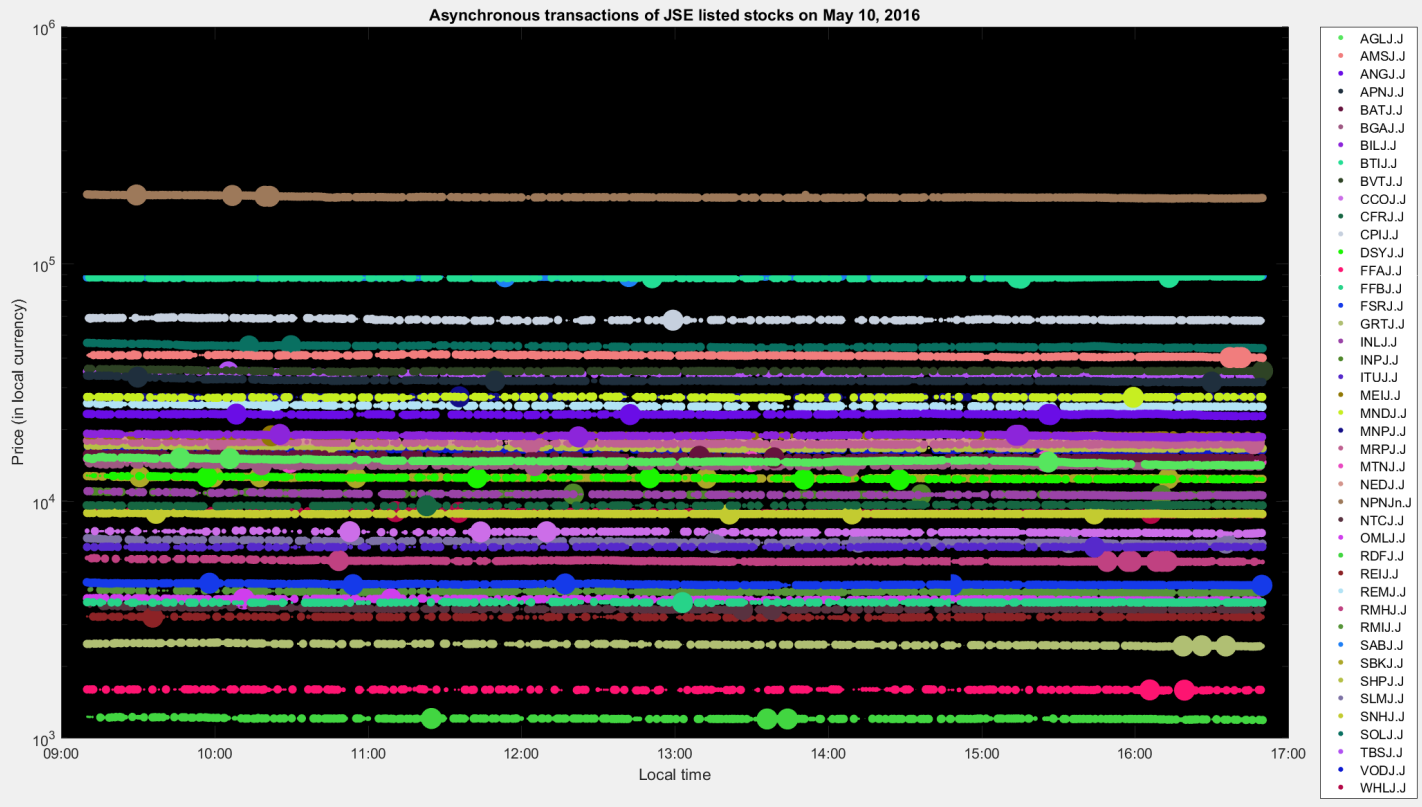


Figure 3.28: Transaction prices for JSE listed stocks on November 04, 2010. Each dot represents a transaction, and the size of the dot is proportional to the size of the transaction. The function provided in appendix C.7 was used to generate the scatter plot of each stock. The test script for this function is provided in appendix C.8.

Price impact of JSE Top 40 constituents

Figure 3.29 visualises the January 2010 to May 2016 average price impact functions for constituents of the JSE Top40. We note that most of the considered stocks have a price impact function that initially decreases, and then increases for large values of ω. Furthermore, from figure 3.30, we note that stocks with a relatively large tend to experience a small price impact for large values of ω.

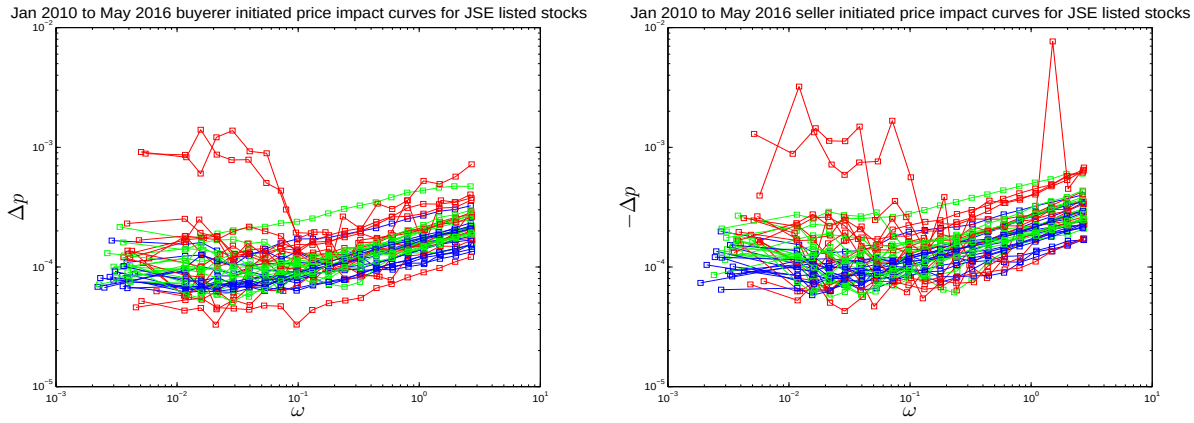


Figure 3.29: January 2010 to May 2016 price impact curves for constituents of the JSE Top40. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

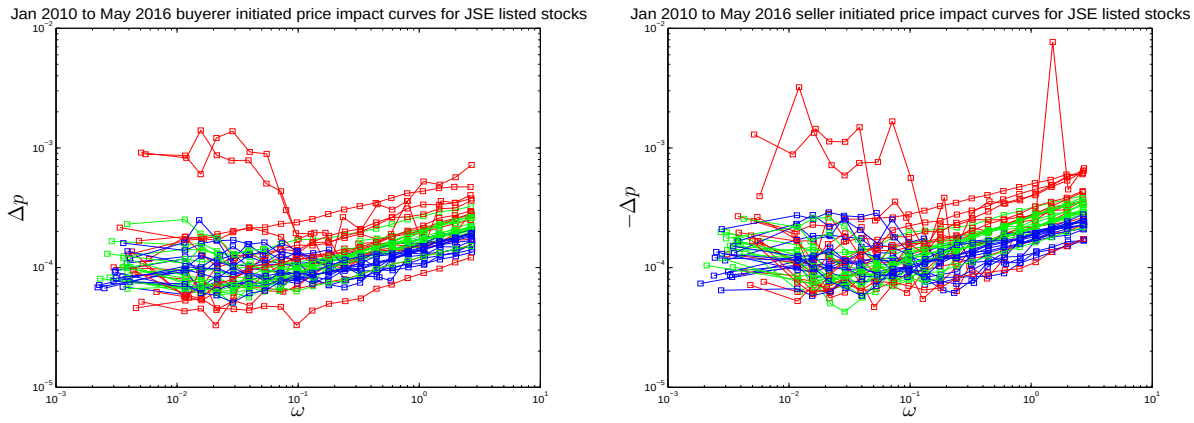


Figure 3.30: January 2010 to May 2016 price impact curves for constituents of the JSE Top40. Stocks were grouped into 3 different average daily volume (ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

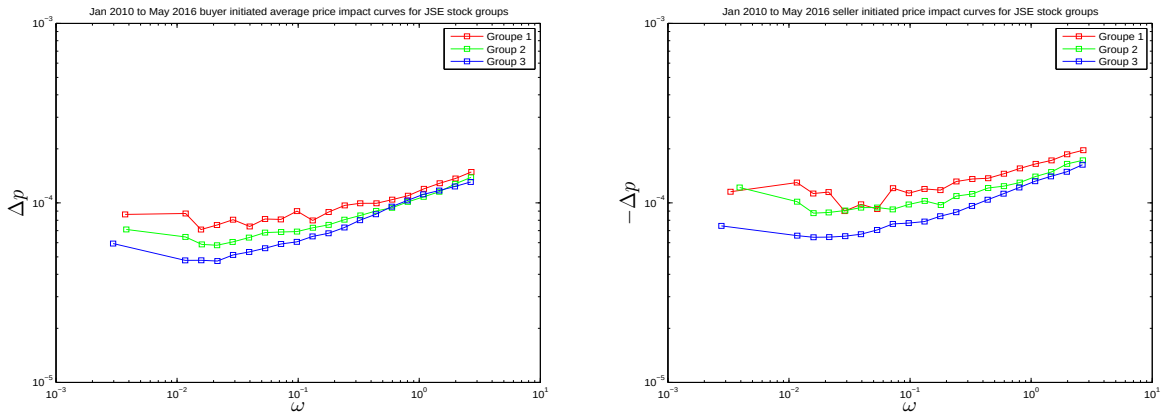


Figure 3.31: January 2010 to May 2016 average price impact curves for constituents of the JSE Top40. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. The average price impact was then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	α_b	λ_b	α_s	λ_s
Group 1	9.1877×10^{10}	1.9393×10^{7}	0.0974	8.7522×10^{3}	0.0931	6.4551×10^{3}
Group 2	1.3580×10^{11}	1.3834×10^{7}	0.1276	9.6878×10^{3}	0.0866	7.5588×10^{3}
Group 3	4.0624×10^{11}	2.4874×10^{7}	0.1673	9.9782×10^{3}	0.1487	8.2908×10^{3}

Table 3.7: Calibrated price impact parameters for JSE stock groups.

After grouping the considered stocks using the method described in section 2.4.3, we constructed three stocks groups, labelled *Group 1* to 3 respectively, and calculated the average price impact for each stock group. Figure 3.31 visualises the calculated price impacts and table 3.7 lists the ADTV and ADV of each stock group. We note that the calculated price impacts are arranged from top to bottom in increasing order of ADTV. Thus, stocks with a relatively high ADTV tend to experience a smaller price impact for most values of ω.

After fitting a relationship of the form

$$\ln \Delta p = \alpha_j \ln \omega - \ln \lambda_j \tag{3.6}$$

to the measured price impacts, we obtained the parameters listed in table 3.7. We note that for both buyer and seller initiated price impacts, λ_j increases with ADTV. Furthermore, we note that the slope of the measured price impact curves α_j tends to increase with ADTV.

Trade sign autocorrelation and price dispersion

Figure 3.32 visualises the January 2010 to May 2016 measured daily average trade sign autocorrelation $\mathcal{C}(\ell)$ and daily average normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the JSE Top 40 index. We note that most of the considered stocks have a trade autocorrelation $\mathcal{C}(\ell)$ that remains highly persistent, even after a large number of trades. Furthermore, we note that all of the considered stocks have a normalises price dispersion $\mathcal{D}(\ell)$ that exhibits sub-diffusion, implying that the price changes, of each considered stocks, are not correlated.

The plot on the left in figure 3.33 does not show a clear relationship between ADV and $\mathcal{C}(\ell)$. This is in contrast to what we observed for B3, BSE and EGX listed stocks in the previous sections. For B3 and ESE stocks, we found that $\mathcal{C}(\ell)$ tends to increase as ADV increases. And for BSE listed stocks, we found that $\mathcal{C}(\ell)$ tends to decrease as ADV increases. However, from the plot on the right in figure 3.33, we note that JSE listed stocks with a relatively high ADV tend to have a smaller normalised price dispersion $\mathcal{D}(\ell)/\ell$. This is consistent with what we observed for B3, BSE and EGX listed stocks.

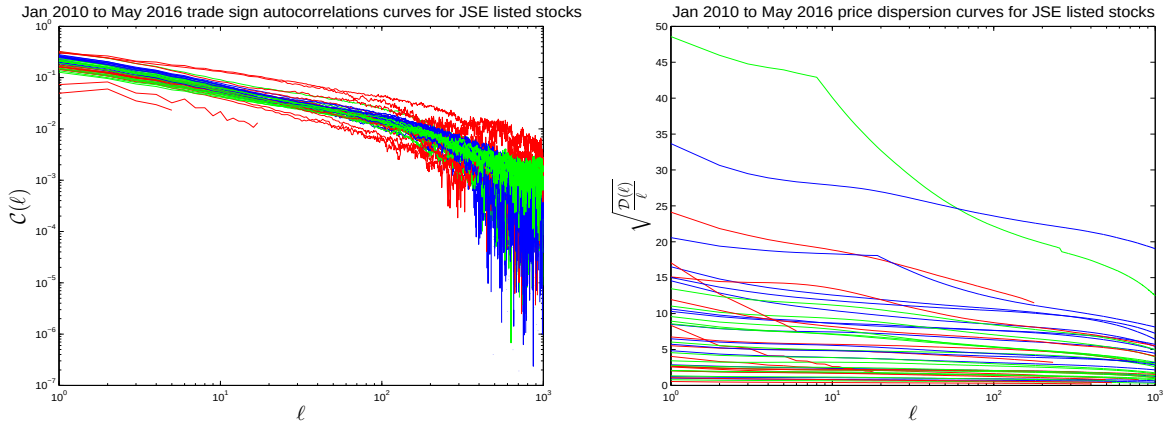


Figure 3.32: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the JSE Top 40. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

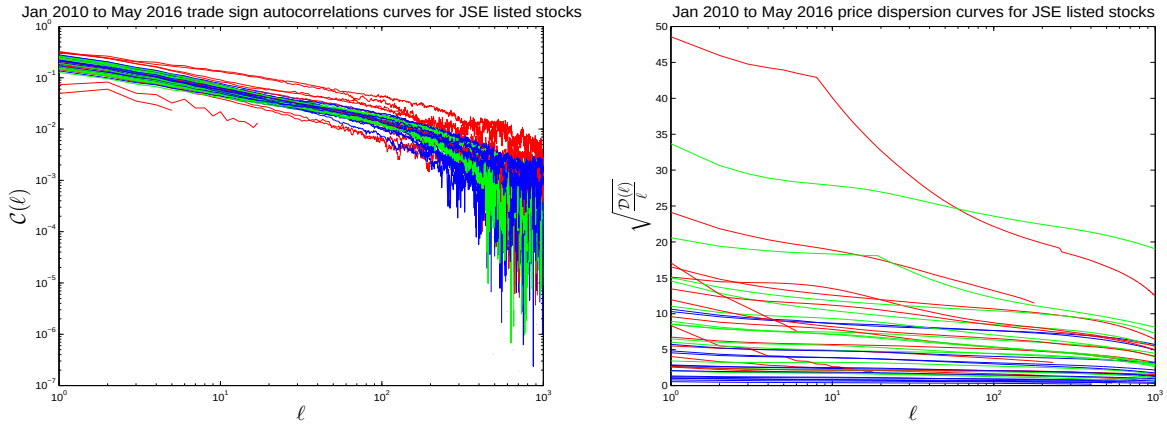


Figure 3.33: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the JSE Top 40. Stocks were grouped into 3 different average daily volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/.

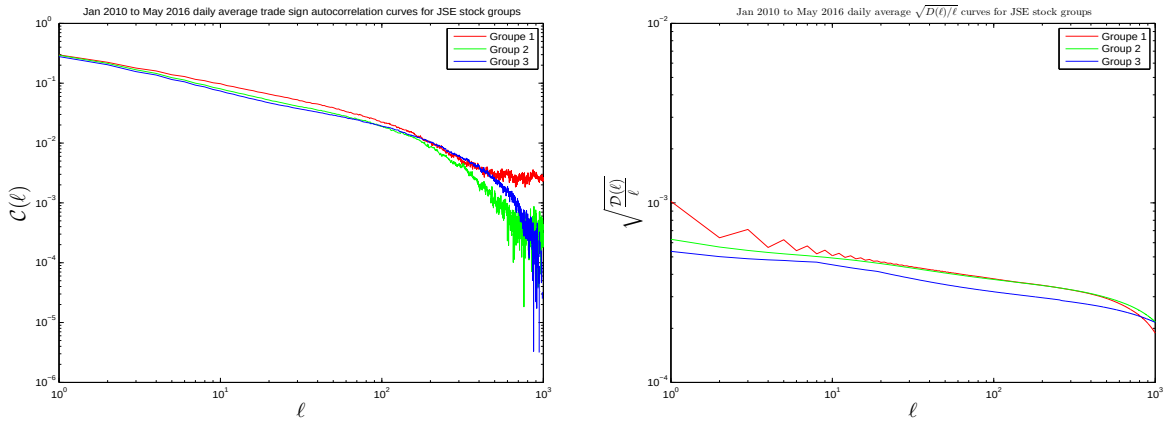


Figure 3.34: January 2010 to May 2016 daily average trade autocorrelation curves $\mathcal{C}(\ell)$ (left) and daily average normalised price dispersion curves $\sqrt{\mathcal{D}(\ell)/\ell}$ (right) for constituents of the JSE Top 40. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository `https://github.com/Telmakaza/Calculate_PriceResponse`.

Stock group	ADVT	ADV	$\mathcal{C}_0$	γ
JSE Group 1	9.1877×10^{10}	1.9393×10^{7}	0.3514	0.6294
JSE Group 2	1.3580×10^{11}	1.3834×10^{7}	0.3362	0.6729
JSE Group 3	4.0624×10^{11}	2.4874×10^{7}	0.3112	0.6557

Table 3.8: Calibrated $\mathcal{C}(\ell)$ parameters for JSE stocks groups

Figure 3.34 visualises the measure $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ for stock groups 1 to 3. After fitting a relationship of the form

$$\mathcal{C}(\ell) = \mathcal{C}_0 \ell^{\gamma} \tag{3.7}$$

to the measured trade sign autocorrelation functions $\mathcal{C}(\ell)$, we obtained the parameters listed in table 3.8. From figure 3.34 and table 3.8, we note that although no clear relationship can be established between $\mathcal{C}(\ell)$ and ADV, as ADTV increases, the value of $\mathcal{C}_0$ decreases, corresponding to a vertical shift $\mathcal{C}(\ell)$. This can be seen clearly in figure 3.34. However, we note that this relationship does not necessarily hold for all ℓ.

Turning our attention to the calibrated γ, we note that $0 < \gamma < 1$, for all of the stock groups. This may imply that the trade sign autocorrelation function $\mathcal{C}(\ell)$ of most the considered stocks is a long memory process. However, as we observed with the previous stock exchanges, figure 3.34 shows that the price changes for all of the considered stock groups are not correlated. In the framework of [29], this may suggest that all of the considered stocks have a bare response like a power law with time.

Price response coefficient and bare response

Figure 3.35 visualises the calculated, January 2010 to May 2016, daily average price response coefficient $\mathcal{R}(\ell)$ and daily average bare response function $G_0(\ell)$ for constituents of the JSE Top 40. From figure 3.35 we note that all of the considered stocks have $\mathcal{R}(\ell)$ that initially increases for small ℓ, and then decreases for large values of ℓ. Furthermore, we note that most of the considered stocks have a $G_0(\ell)$ that decreases like a power law.

Turning our attention to figure 3.36, we note that there is a visible relationship between price response and ADV: stocks that have a high ADV tend to have a lower price response, where price response is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$.

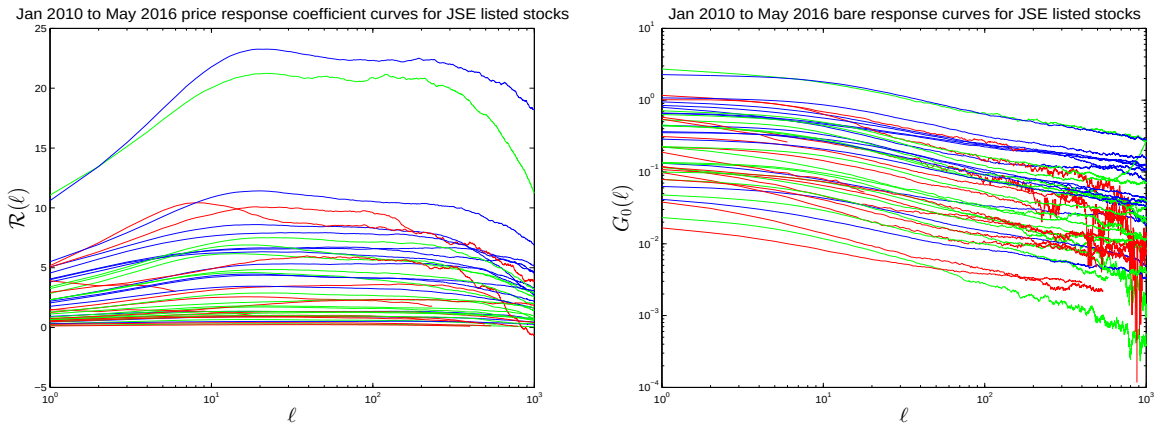


Figure 3.35: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the JSE Top 40. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

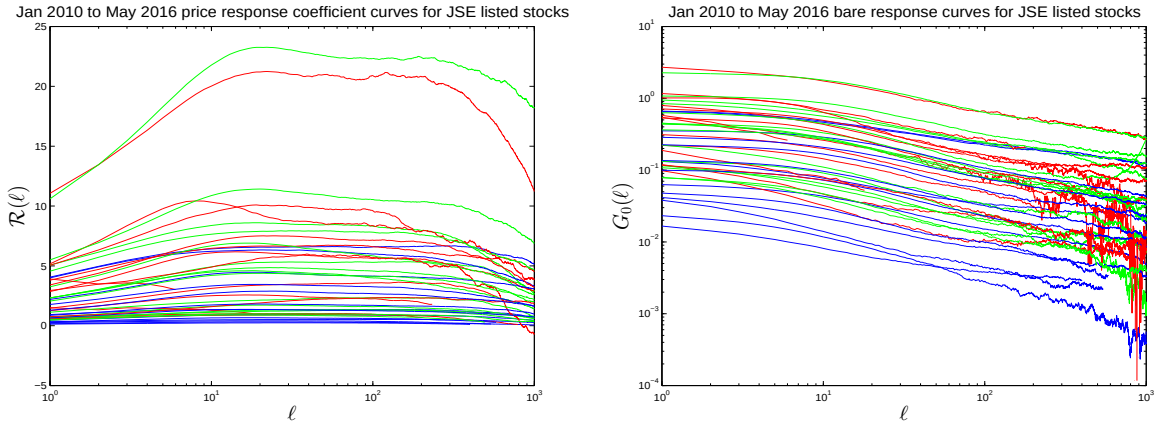


Figure 3.36: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the JSE Top 40. Stocks were grouped into 3 different average daily traded volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

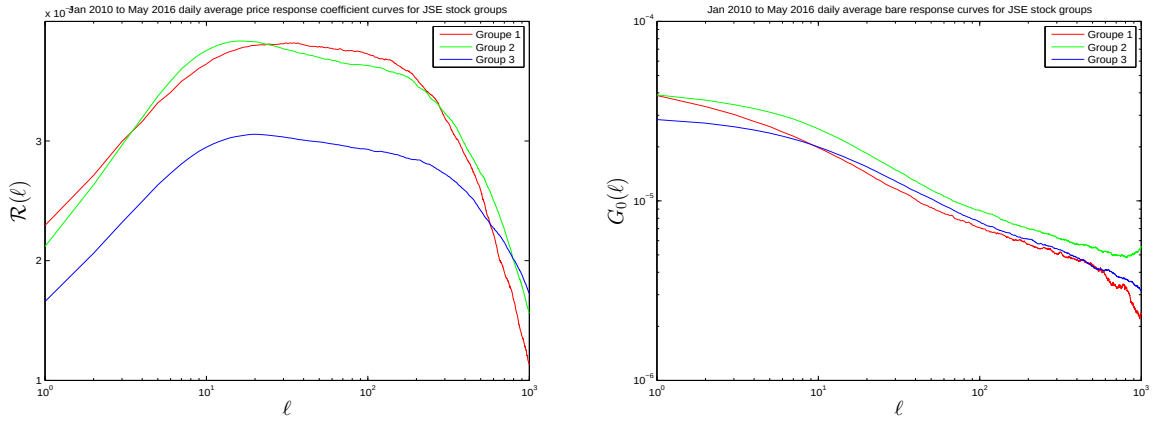


Figure 3.37: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ (left) and daily average bare response curves $G_0(\ell)$ (right) for constituents of the Bovespa Index. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{R}(\ell)$ and $G_0(\ell)$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	Γ_0	ℓ_0	β
Group 1	9.1877×10^{10}	1.9393×10^{7}	4.4306×10^{-5}	0	0.3875
Group 2	1.3580×10^{11}	1.3834×10^{7}	4.7733×10^{-5}	0	0.3480
Group 3	4.0624×10^{11}	2.4874×10^{7}	3.8792×10^{-5}	0	0.3480

Table 3.9: Calibrated $G_0(\ell)$ parameters for JSE stocks groups

In order to further investigate the aggregate behaviour of stocks, we calculated $\mathcal{R}(\ell)$ and $G_0(\ell)$ for stock groups 1 to 3. Figure 3.37 visualises the results. From these figures alone, it is difficult to see the relationship between price response and ADV that we saw in figure 3.36. Thus, we fit a relationship of the form

$$G_0(\ell) = \frac{\Gamma_0}{(\ell_0 + \ell)^{\beta/2}} \tag{3.8}$$

to the calculated price response functions. Table 3.9 lists the calibrated parameters. Although no clear relationship can bet established between price response and ADTV, we note that as ADV increases, Γ_0 decreases, corresponding to a downward shift in $G_0(\ell)$.

Concluding remarks

This section considered the price response of the constituents of the JSE Top 40 Index. We started by describing the stock exchange at which these stocks are listed (the JSE), and then we gave an overview of the JSE Top 40 index. We then considered the January 2010 to May 2016 average price impact $\mathcal{R}(\ell = 1, V)$, daily average price trade sign auto-correlation $\mathcal{C}(\ell)$, daily average normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$, daily average price response coefficient $\mathcal{R}(\ell)$ and daily average bare response $G_0(\ell)$. We found that most of the considered stocks have an average price impacts $\mathcal{R}(\ell = 1, V)$ that first initially

decrease for small ω and then increase like a power law for large ω. We also found that the considered stocks have a highly persistent trade sign autocorrelation process $\mathcal{C}(\ell)$, a price dispersion $\mathcal{D}(\ell)$ process that exhibits sub-diffusion, price response coefficients that initially increase for small small ℓ and then decrease for large values of ℓ, and bare response functions $G_0(\ell)$ that decrease like a power law.

We also found that there is a discernible relationship between ADTV and $\mathcal{R}(\ell = 1, V)$, $\mathcal{D}(\ell)$ and $\mathcal{C}(\ell)$. In particular, we found that groups stocks that have a relatively high ADTV tend to have a small price impact for most *omega*. Furthermore, we found that the $\mathcal{C}(\ell)$ and $\mathcal{D}(\ell)$ of the considered stock groups arrange themselves from top to bottom in increasing order of ADTV.

Lastly, we also observed a discernible relationship between ADV and $\mathcal{R}(\ell)$ and $G_0(\ell)$. Moreover, we found that stocks that have a low ADV tend to have small value of $\mathcal{R}(\ell)$ and $G_0(\ell)$.

3.5 Moscow Exchange

The Moscow Exchange (MOEX) is an exchange group in Russia that a offers trading services in stocks, bonds, derivatives and other financial instruments. As of June 2017, the exchange has more than 200 listed companies and a market capitalization just above $USD 500 billion. The MOEX is relatively young (at least compared to the other markets that we consider in this dissertation). It was established in 2011, after a merger between the Moscow Interbank Currency Exchange (MICEX) and the the Russian Trading System (RTS) [78]. MICEX was established on the 9^{th} of January 1992. The exchange launched futures trading in 1996, online trading in 1999 and received full member status of the Word Federation of Exchanges in 2009, before merging with RTS two years later. On the other hand, the RTS was established 1995, however, its roots can be traced back to 1994 when a group of 15, Moscow based brokerages formed the Professional Association of Stock Market Participants (PAUFOR) [79]. With help from NASDAQ and an American computer company, the brokers established a NASDAQ-like over-the-counter computer network, upon which the RTS was lauched [79].

MOEX equity market model

Each equity trading session on the Moscow Exchange is made up of three main periods: An *opening auction*, a *main trading period* and a *closing auction*. The opening auction commences at 09:50:00 and ends at 09:59:59. Both Limit and market orders are permissible during this period, however, market orders have the highest priority [80]. The main trading period (for equities) starts at 10:00:00 and concludes at 18:39:59. Market, limit and iceberg orders can be submitted during this period. Further, trading members can access information about their orders, and 20 orders with the best price [81]. When the main trading period concludes, unexecuted limit orders are carried over the the closing auction period. The closing auction starts at 18:40:01 and ends at 18:50:00 [82]. During this period, the representative closing price is determined. Orders are collected from 18:40:01 to 18:45(00-29), and the period from 18:45:00 to 18:45:29 is used to determine the closing price for the execution of orders. Lastly, trades are executed from 18:45:00 to 18:50:00. For more information on the equity market structure of the MOEX, refer to [80], [81] and [82].

RTS index

This study considers stocks that constitute the RTS index. The RTS Index is a capitalization-weighted composite index calculated based on the prices of the 50 most liquid MOEX listed stocks. Appedix A.5 lists the constituents of the RTS index from January 2010 to May 2016. For more information regarding the RTS Index and other benchmark indices on the MOEX, refer to [83]. Figure 3.38 illustrates the transactions prices and volumes of some constituents of the RTS Index on the 4^{th} of May 2016. Below, we consider the price response of trades for these stocks.

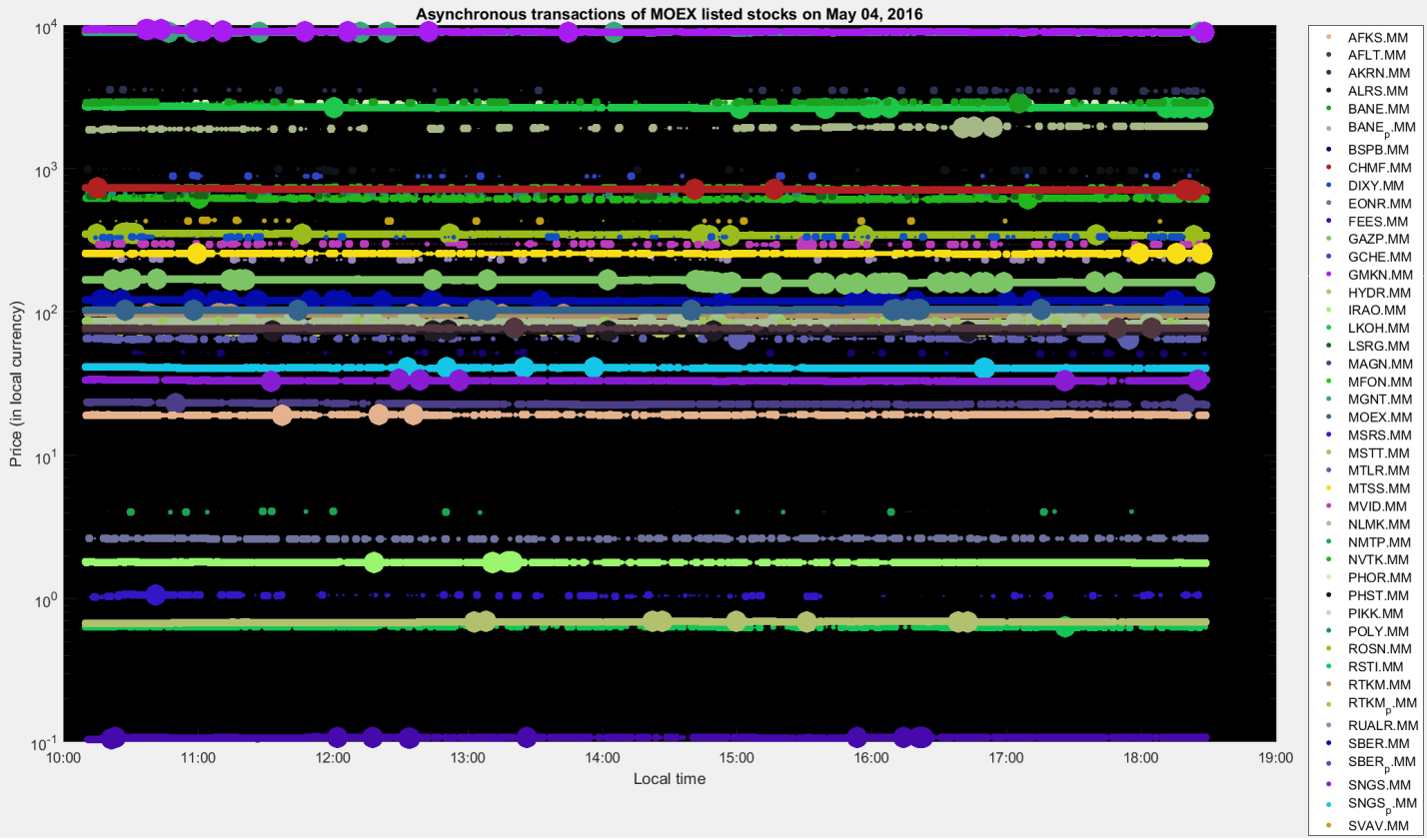


Figure 3.38: Transaction prices for MOEX listed stocks on May 04, 2016. Each dot represents a transaction, and the size of the dot is proportional to the size of the transaction. The function provided in appendix C.7 was used to generate the scatter plot of each stock. The test script for this function is provided in appendix C.8.

Price impact curves for constituents of the RTS index

Figure 3.39 visualises that calculated average price impacts for stocks that constitute the RTS index. We note that most of the considered stocks have a price impact functions that increase like a power law for all ω. Furthermore, figure 3.39 shows that stocks that have a relatively high ADTV tend to experience a smaller magnitude of price impact, for both buyer and seller initiated trades. Lastly, figure 3.40 illustrates the relationship between ADV and average price impact. We again, we note that stocks have a relatively high ADV tend to have a relatively small price impact.

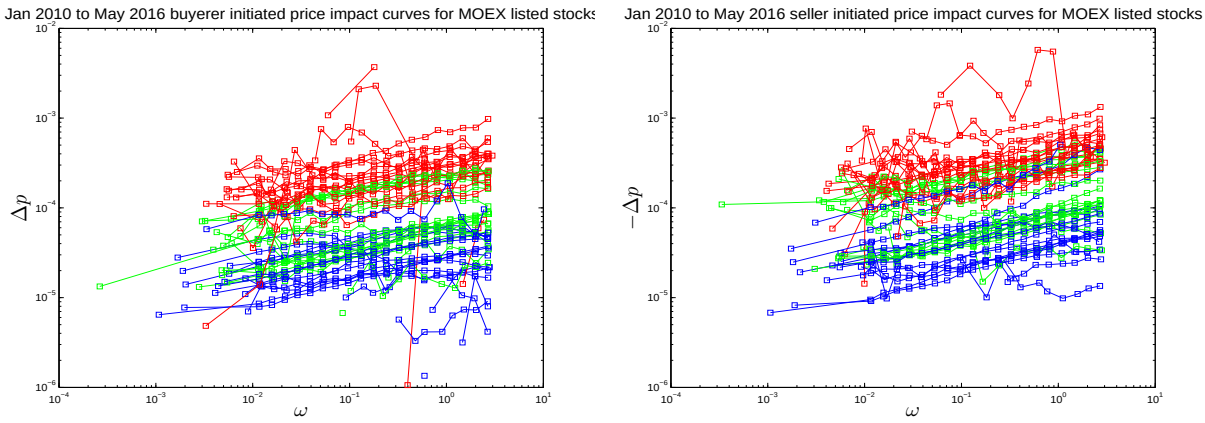


Figure 3.39: January 2010 to May 2016 price impact curves for constituents of the RTS index. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

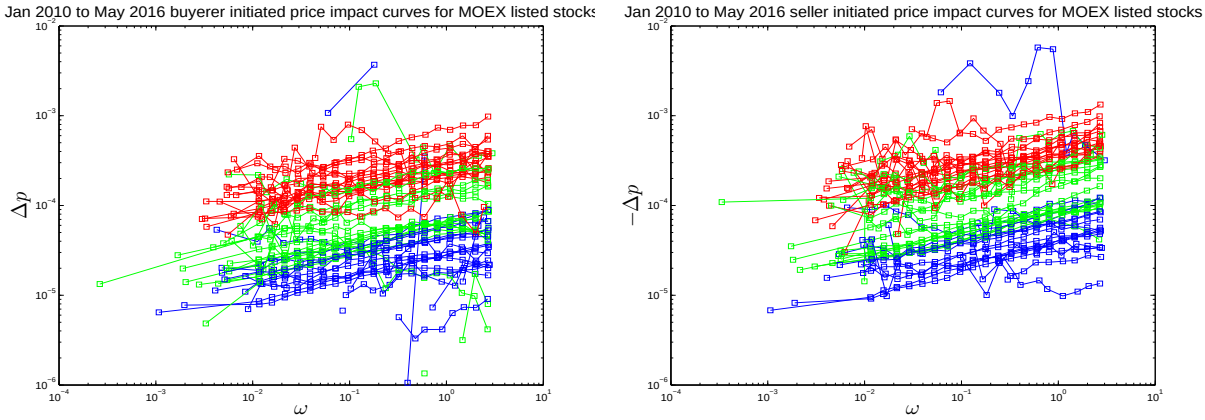


Figure 3.40: January 2010 to May 2016 price impact curves for constituents of the RTS index. Stocks were grouped into 3 different average daily volume (ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse

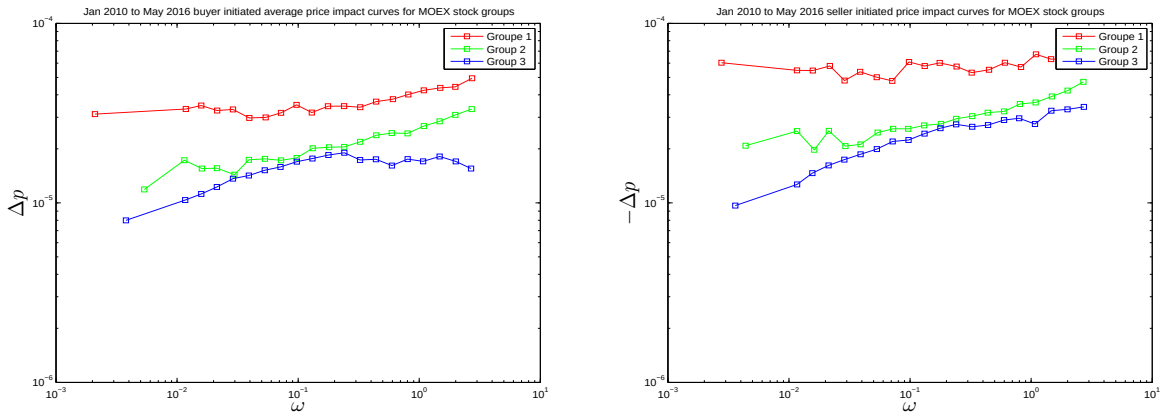


Figure 3.41: January 2010 to May 2016 average price impact curves for constituents of the RTS index. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. The average price impact was then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	α_b	λ_b	α_s	λ_s
Group 1	3.1304×10^9	4.4597×10^9	0.0591	2.4957×10^4	0.0278	1.6538×10^4
Group 2	4.1752×10^9	6.6207×10^9	0.1421	3.7657×10^4	0.1243	2.7629×10^4
Group 3	1.2043×10^{10}	1.3051×10^{10}	0.0955	5.5040×10^4	0.1766	3.1792×10^4

Table 3.10: Calibrated price impact parameters for MOEX stock groups.

In order to investigate the price impact of groups of stocks, we binned the stocks into three different groups, labelled *Group* 1 to 3 respectively. Figure 3.41 visualises the calculated price impact for each stock group and table 3.10 lists the ADTV and ADV of each stock group. As we saw in figure 3.39 and 3.40, stocks that have a relativity high ADV and ADTV tend to experience a comparably smaller magnitude of price impact for all ω. After fitting a relationship of the form

$$\ln \Delta p = \alpha_j \ln \omega - \ln \lambda_j \tag{3.9}$$

to the calculated price impacts, we obtained the parameters listed in table 3.10. Here λ_b and α_b are the calibrated parameters for buyer initiated price impacts and λ_s and α_s are those for seller initiated price impacts. We note that as ADTV and ADV increase, λ_j, where $j = \{b, s\}$, also increases — corresponding to a vertical downward shift in price impact. This observation is consistent with that made in figure 3.39, 3.40 and 3.41.

Trade sign autocorrelation and price dispersion

Figure 3.42 visualises the calculated January 2010 to May 2016 daily average trade sign autocorrelation $\mathcal{C}(\ell)$ and daily average normalised price dispersions $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the RTS index. From the plot on the left in figure 3.42, we note that most of the considered stocks have a trade sign autocorrelation process $\mathcal{C}(\ell)$ that remains highly persistent, even after a large number of trades. Furthermore, from the plot on the right in figure 3.43, we note that all of the considered stocks have a normalised price dispersion process $\sqrt{\mathcal{D}(\ell)/\ell}$ that exhibits sub-diffusion.

Turning our attention to the relationship between ADTV and $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$, from figure 3.42, we note that stocks that have a relatively high ADTV tend to exhibit a relatively high degree of trade sign autocorrelation $\mathcal{C}(\ell)$. However, from the plot on the right in figure 3.43, we note that it is difficult to establish a relationship between ADTV and $\sqrt{\mathcal{D}(\ell)/\ell}$.

Figure 3.43 illustrates is similar to 3.42, however, it illustrates the relationship between ADV and $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$. From the plot on the left in this figure, we note that stocks that have a relatively high ADV tend to have a relatively small value of $\mathcal{C}(\ell)$. (However, one could argue that the relationship between $\mathcal{C}(\ell)$ and ADTV is stronger.) We also note that although no clear relationship could be established between ADTV and $\sqrt{\mathcal{D}(\ell)/\ell}$, one can see a discernible correlation between ADV and the latter: stocks that have a relatively large ADV tend to exhibit a smaller price normalised dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$ for all ℓ.

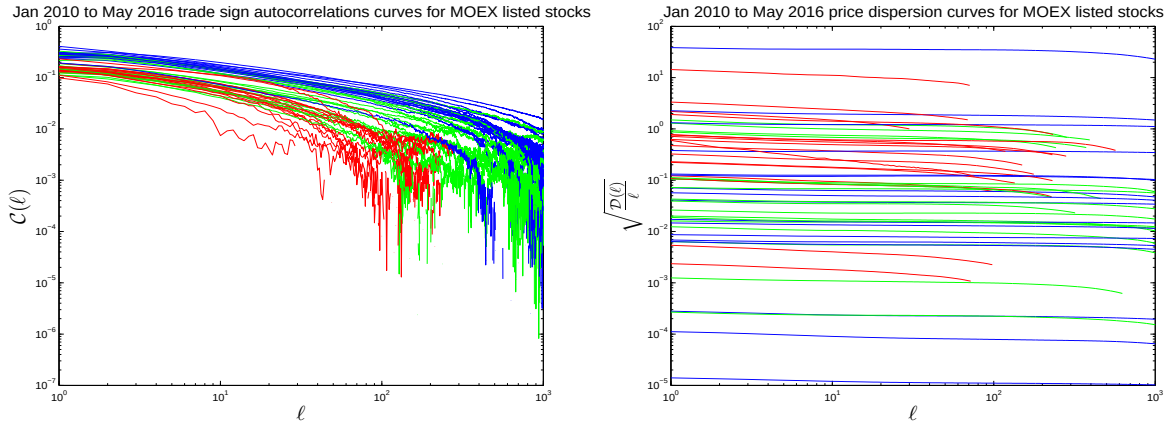


Figure 3.42: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the RTS index. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

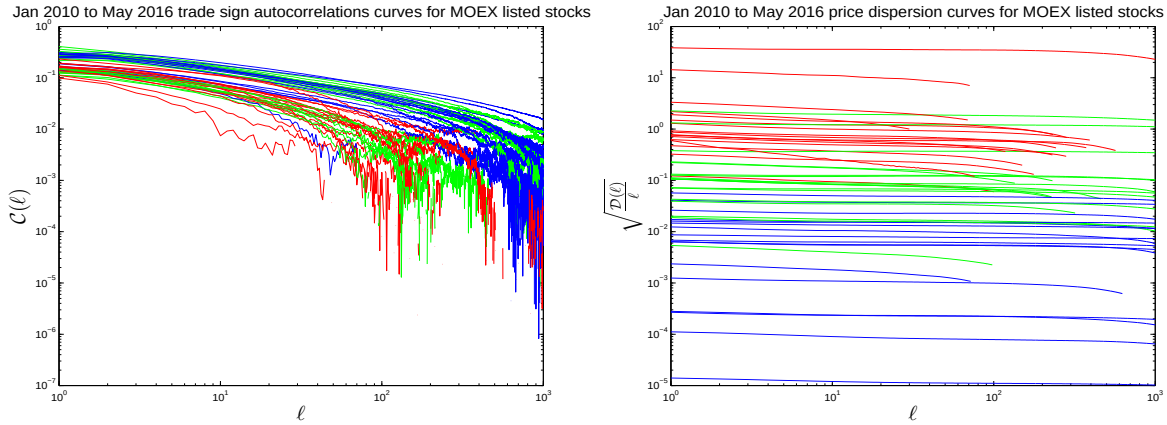


Figure 3.43: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the RTS index. Stocks were grouped into 3 different average daily volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/.

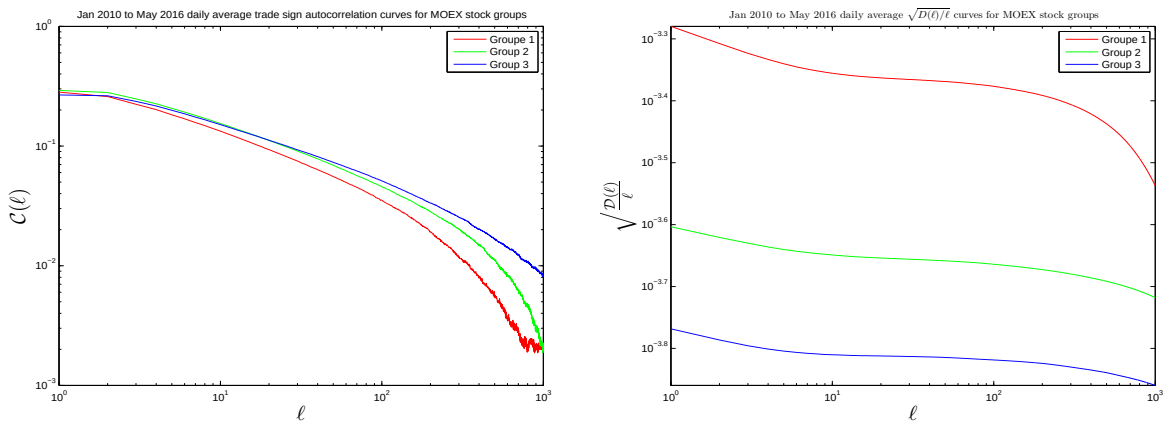


Figure 3.44: January 2010 to May 2016 daily average trade autocorrelation curves $\mathcal{C}(\ell)$ (left) and daily average normalised price dispersion curves $\sqrt{\mathcal{D}(\ell)/\ell}$ (right) for constituents of the RTS index. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	$\mathcal{C}_0$	γ
Group 1	3.1304×10^9	4.4597×10^9	0.3986	0.5705
Group 2	4.1752×10^9	6.6207×10^9	0.4291	0.5256
Group 3	1.2043×10^{10}	1.3051×10^{10}	0.3911	0.4736

Table 3.11: Calibrated $\mathcal{C}(\ell)$ parameters for MOEX stocks groups

In order to further investigate $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ for different stock groups, we calculated these quantities for stock groups 1 to three. Figure 3.44 visualises the calculated results. As with the observations made in figure the plots on the left in figure 3.42, we find that stocks that have a relatively high ADV and ADTV tend to exhibit a high degree of trade sign autocorrelation $\mathcal{C}(\ell)$. However, in figure 3.44, this observation is only made for large values of ℓ.

After fitting a relationship of the form

$$\mathcal{C}(\ell) = \mathcal{C}_0 \ell^{\gamma} \tag{3.10}$$

to the measured trade sign autocorrelation functions $\mathcal{C}(\ell)$, we obtained the parameters listed in table 3.11. We note that for all of the constructed stock groups $0 < \gamma < 1$. This may suggest that the considered stocks have a long memory trade sign autocorrelation process. However, as can be seen from the plot on the left in figure 3.44, we note that although the considered stocks may have a long memory trade sign autocorrelation process, they do not have price changes that are correlated. In the frame work of [29], this may suggest that the considered stocks have bare response that decreases like a power law. We consider the empirical bare response of these stocks in the section below.

Price response coefficient and bare response

Figure 3.45 visualises the calculated $\mathcal{R}(\ell)$ and $G_0(\ell)$ for each of the considered stocks. All of the considered stocks have a $\mathcal{R}(\ell)$ that initially increases for small ℓ and then decreases for large values ℓ. Furthermore, we note that with the exception of a few stocks [4], most stocks have a $G_0(\ell)$ that decrease with ℓ. Turning our attention to figure 3.46, we note that stocks that have a relatively relatively high ADV tend to have relatively low price response, where the latter is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$. Furthermore, we note that it is precisely the stocks that have a relatively low ADV that tend to have a $G_0(\ell)$ that increases.

[4] We are unclear as to why some stocks have a $G_0(\ell)$ that increases, but we speculate that it may be due errors in the calculation and/or the data.

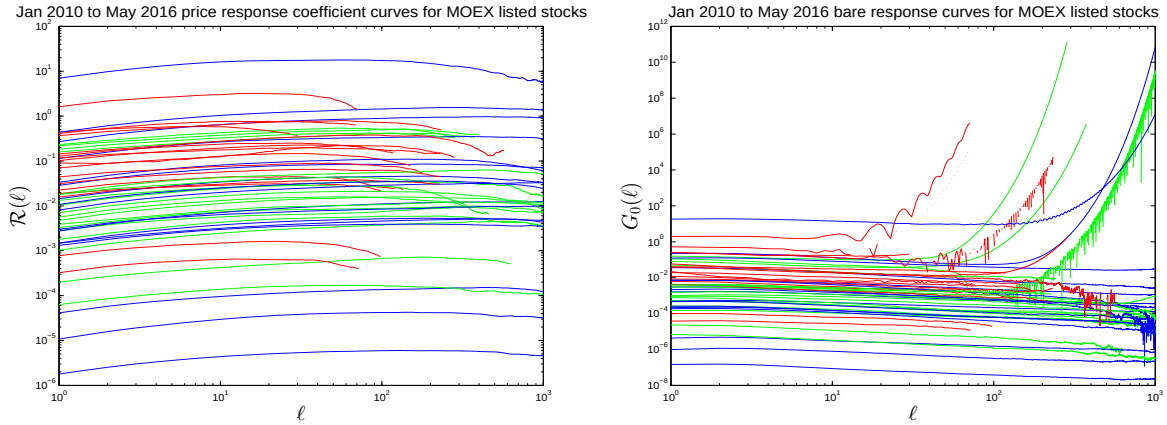


Figure 3.45: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the RTS index. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

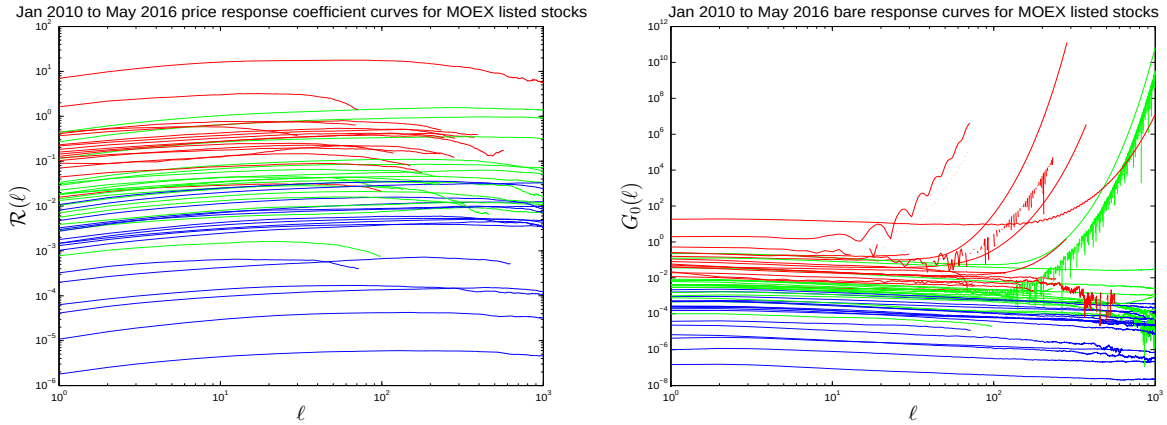


Figure 3.46: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the RTS index. Stocks were grouped into 3 different average daily traded volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

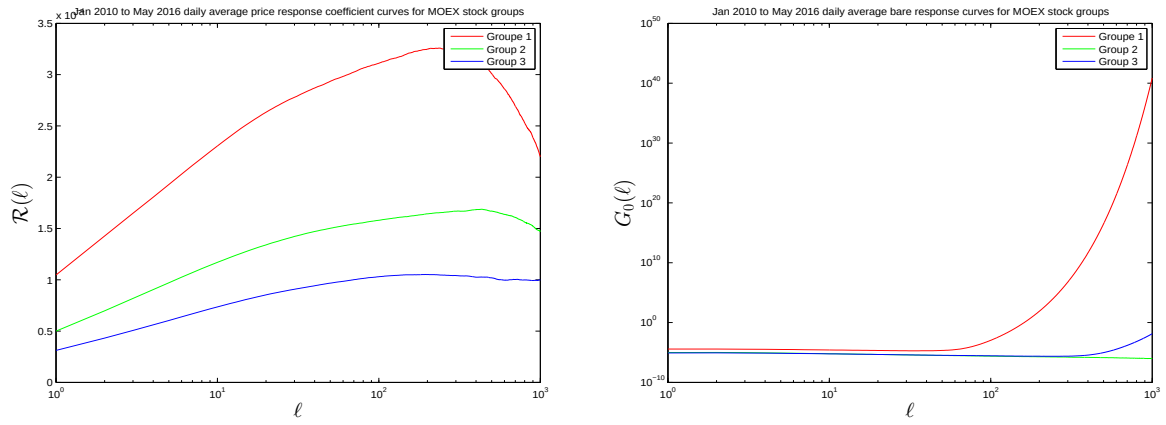


Figure 3.47: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ (left) and daily average bare response curves $G_0(\ell)$ (right) for constituents of the RTS Index. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{R}(\ell)$ and $G_0(\ell)$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	Γ_0	ℓ_0	β
Group 1	3.1304×10^9	4.4597×10^9	0	0	0
Group 2	4.1752×10^9	6.6207×10^9	1.2957×10^{-5}	0	0.3624
Group 3	1.2043×10^{10}	1.3051×10^{10}	6.2056×10^{-8}	0	-1.4911

Table 3.12: Calibrated $G_0(\ell)$ parameters for JSE stocks groups

In order to further investigate the $\mathcal{R}(\ell)$ and $G_0(\ell)$ of stock groups, we calculated the former and latter for stock groups 1 to three. The results that we obtained are shown in figure 3.47. The plot on the left in figure 3.45 shows that stocks that have a relatively high ADV and ADTV tend have a lower price response coefficient.

Concluding remarks

This section considered the price response of stocks that constitute the RTS index. We started by describing the stock exchanges at which these stocks are listed (the MOEX) and then we gave a brief overview of the RTS index. We then considered the average price impact of trades for these stock. We found that most of the considered stocks have a price impact function that increases like a power law for all ω. Furthermore, we found that stocks that have a relatively high ADTV and ADV tend to exhibit a smaller price impact. We then investigated the daily average trade sign autocorrelation $\mathcal{C}(\ell)$ and daily average normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$ for these stocks. We found that most of the considered stocks have a highly persistent trade sign autocorrelation process $\mathcal{C}(\ell)$ and a normalised price dispersion process $\sqrt{\mathcal{D}(\ell)/\ell}$ that exhibits sub-diffusion. Furthermore, we found that stocks with a relatively high ADTV and ADV tend to a relatively high degree of trade sign autocorrelation $\mathcal{C}(\ell)$. We were also observed a strong relationship between $\sqrt{\mathcal{D}(\ell)/\ell}$: stocks with a high ADV tend to have a smaller value of $\sqrt{\mathcal{D}(\ell)/\ell}$ for

all ℓ. Lastly, we investigated the price response coefficient and bare response function for the considered stocks. We found that all of the considered stocks have a $\mathcal{R}(\ell)$ that initially increases for small values of ℓ and then decreases for large ℓ. Furthermore, we found most of the considered stocks have a $G_0(\ell)$ that decreases with ℓ. Lastly, we found that stocks that have a relatively large ADV tend to have a smaller price response for all ℓ, where price response is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$.

3.6 Nairobi Securities Exchange

The Nairobi Securities Exchange (NSE) was officially inaugurated in 1954. Prior to that, the exchange consisted of dealings between participants taking place on a *gentleman's agreement*. However, in 1953, the London Stock Exchange (LSE) accepted to recognize the setting up of the NSE as an overseas stock exchange, and between 1954 and 1962, the NSE was registered under the Societies Act (1954) [84]. In 1994, the NSE was rated as the best performing market in the world when the NSE 20-share index reached a record high of 5030 points. The exchange also moved to a new location and established a computerized delivery and settlement system in the same year. February 1, 2008 saw the exchange extending it's trading hours to 7 hours per day. Before that, the exchange only conducted two hour trading sessions prior to September 11, 2006, and three hour sessions from there onward until February 2008. As of August 2017, the NSE has a market cap of close to $USD 25 billion and a total of 64 listed companies[69].

NSE equity market model

Each trading day on the NSE starts at 09:00 and ends at 15:00. Pre-open starts at 09:00 and ends at 09:30. This is followed by an opening auction at 09:30 and continuous trading from 09:30 to 15:00. Closing is at 15:00. Here, the time is determined by the NSE electronic trading system: Automated Trading System (ATS). Participants can submit market and limit orders to trade securities, and active limit orders are matched by price-time priority. For a more detained consideration of NSE equity market trade rules, refer to [85]

NSE20 index

This study considers stocks that constitute the NSE20 index. The NSE20 is a 20 share benchmark index in the NSE that is made up of "blue chip" stocks that meet a specific criteria. Refer to [86] for the selection criteria list. Appendix A.6 lists the constituents of this index from January 2010 to May 2016. Figure 3.48 illustrates the transactions prices and volumes of NSE20 constituents on the 4^{th} of May 2016. Below, we consider the price response of trades for these stocks.

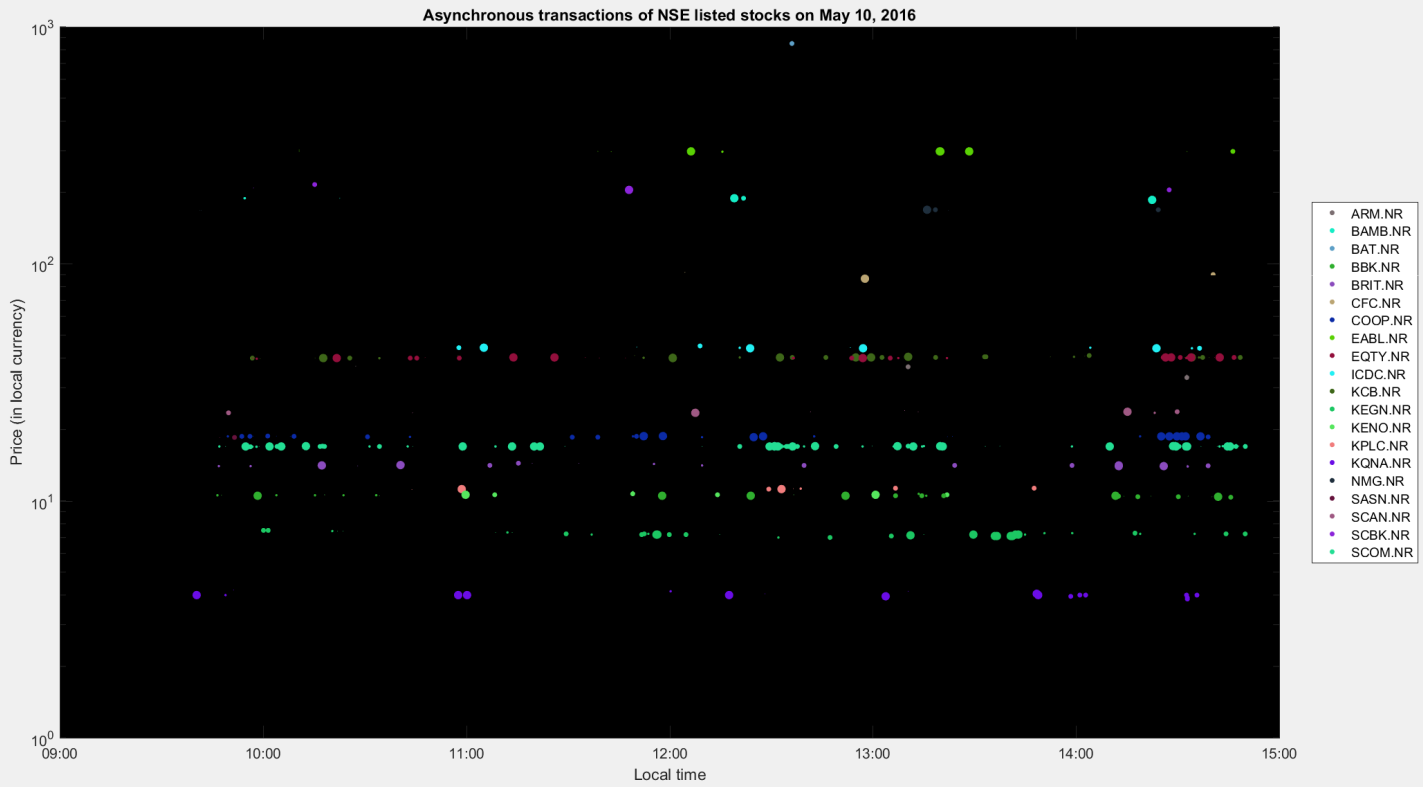


Figure 3.48: Transaction prices for NSE listed stocks on May 04, 2016. Each dot represents a transaction, and the size of the dot is proportional to the size of the transaction. The function provided in appendix C.7 was used to generate the scatter plot of each stock. The test script for this function is provided in appendix C.8.

Price impact curves for constituents of the NSE20

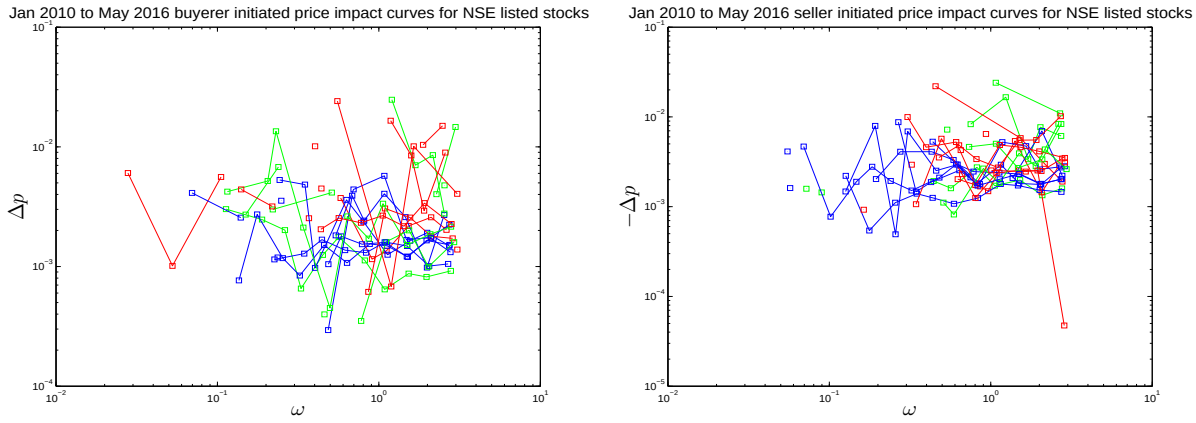


Figure 3.49: January 2010 to May 2016 price impact curves for constituents of the NSE20. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

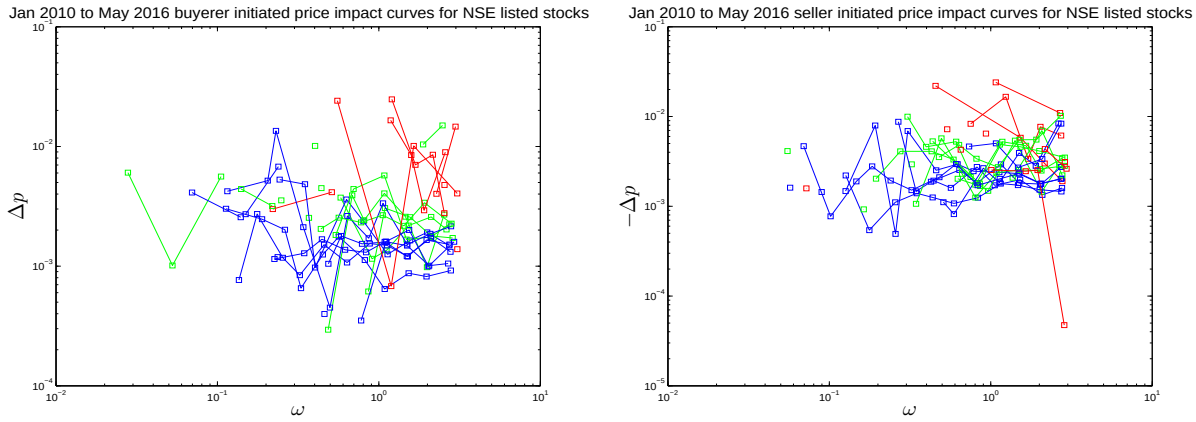


Figure 3.50: January 2010 to May 2016 price impact curves for constituents of the NSE20. Stocks were grouped into 3 different average daily volume (ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script to reproduce all of these figures can be found at https://github.com/Telmakaza.

Figure 3.49 and 3.50 visualise the calculated price impacts of trades for stocks that constitute the NSE20. We note that the considered stocks have price impacts that are not similar to those observed in previous sections: the average price change Δp does not appear to form a power law relationship with the transacted normalised volume ω. Furthermore, we not that it is difficult to establish a relationship between price impact $\mathcal{R}(\ell = 1, V)$ and ADTV or ADV.

Trade sign autocorrelation and price dispersion

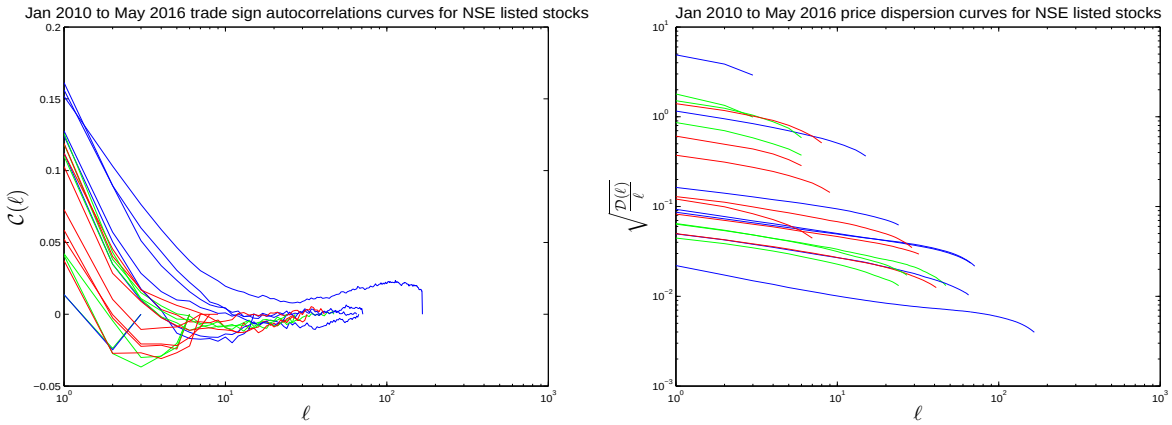


Figure 3.51: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the NSE20. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

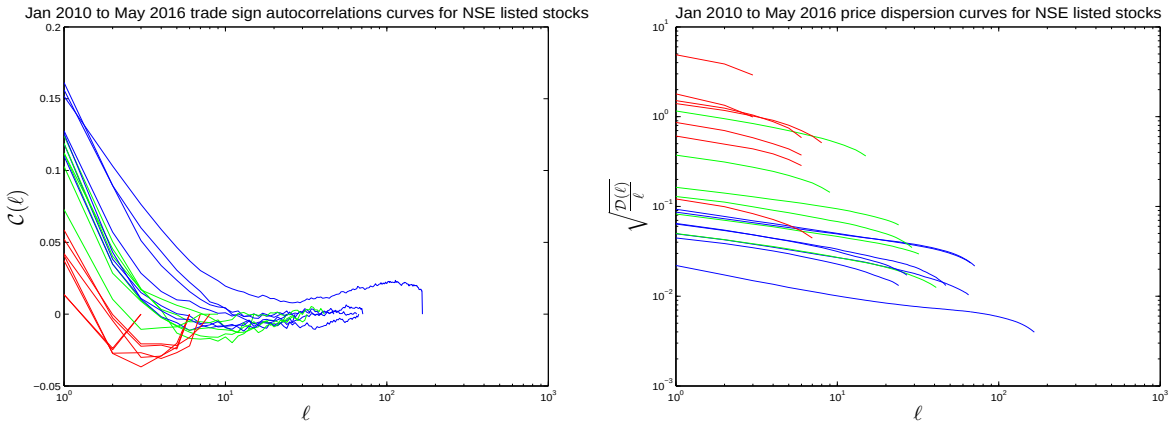


Figure 3.52: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the NSE20. Stocks were grouped into 3 different average daily volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/.

Figure 3.51 visualise the calculated January 2010 to May 2016 average daily average trade sign autocorrelation $\mathcal{C}(\ell)$ and daily average normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$. We note that all of the considered stocks have a $\mathcal{C}(\ell)$ that appears to be less persistent than what has been observed for stocks in markets like the JSE and BSE. However, we note that most of the considered stocks in this exchange have a $\sqrt{\mathcal{D}(\ell)/\ell}$ that is similar to what

has been observed in the markets considered in the previous section. Furthermore, from figure 3.52, we note that there is a discernible relationship between ADV and $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$: stocks that have a relatively high ADV appear to have a smaller value of $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ for all ℓ.

Price response coefficient and bare response

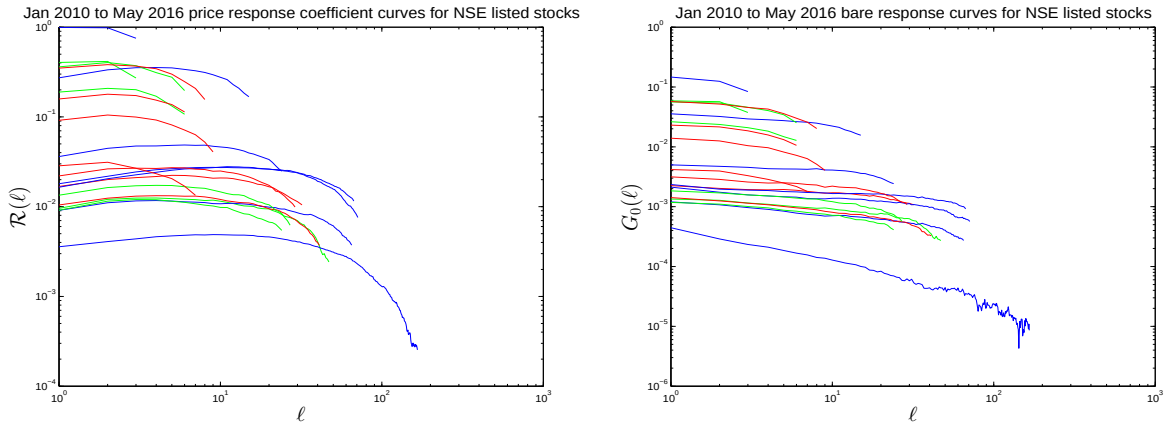


Figure 3.53: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the NSE20. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

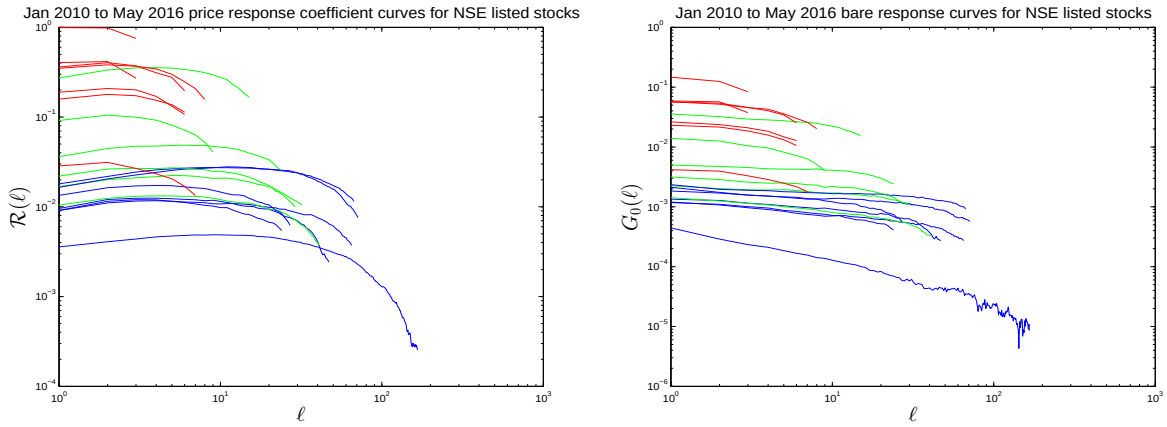


Figure 3.54: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the NSE20. Stocks were grouped into 3 different average daily traded volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Figure 3.53 visualises the calculated January 2010 to May 2016 daily average price response coefficient $\mathcal{R}(\ell)$ and bare response function $G_0(\ell)$ for constituents of the NSE20. We note that all of the considered stocks have a $\mathcal{R}(\ell)$ that initially increase for small ℓ

and then decreases for large values of ℓ. Furthermore we note that all of the considered stocks have a $G_0(\ell)$ that decreases like a power law.

Turning our attention to figure 3.54, we note that there is a discernible relationship between price ADV and price response: stocks that have a relatively ADV tend to have a relatively small price response, where price response is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$.

Concluding remarks

This section considered the price response of trades for stocks that constitute the NSE20 index. We started by describing the listing exchange of these stocks (the NSE) and then we gave a brief overview of the NSE20 index. We then investigated the January 2010 to May 2016 average price impact for the constituent stocks of this index. We found that the calculated price impact curves did not show the behaviour that we have observed in previously considered markets (like the JSE and BSE). In particular, the price impacts did not show a behaviour of the form $\mathcal{R}(\ell) \propto \omega^{\alpha}$, and no discernible relationship between ADV and ADTV could be established. We then investigated the daily average trade sign autocorrelation $\mathcal{C}(\ell)$ and daily average normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$. We found that the calculated $\mathcal{C}(\ell)$ does not show long range autocorrelations, like in the previously considered markets. However, we found that most stocks have a $\sqrt{\mathcal{D}(\ell)/\ell}$ that exhibits sub-diffusion. Lastly, we observed a discernible relationship between ADV and $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$: stocks with a relatively high ADV tend to have a relatively low $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$.

We then investigated the daily average price response coefficient $\mathcal{R}(\ell)$ and bare response function $G_0(\ell)$. We found that all of the considered stocks have a $\mathcal{R}(\ell)$ that initially increases for small values of ℓ and then decreases for large ℓ. We also observed that all of the considered stocks have a $G_0(\ell)$ that decreases as ℓ increase. Lastly, we found that stocks that have a relatively high ADV tend to experience a relatively low price response, where price response is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$. These findings are consistent with the observations that we made in all of the previously considered markets.

However, given the sparse liquidity in this market, it is difficult to assert these observations. From figure 3.49 and 3.50, we note that the price impact curves for some of the illiquid stocks are broken lines. This is an artefact of the plotting system that we used in MATLAB: price impact values for some volumes are negative and these are not included in the log-log plot. Negative price impact are not what we expected. We conjuncture that the illiquidity of these stocks permitted this phenomena.

3.7 Shanghai Stock Exchange

The Shanghai Stock Exchange (SSE) is one of the largest stock exchange in the world. As of October 2017, it has a total of 1355 listed companies and a market capitalization of $USD 4.9 trillion [69]. The beginnings of the SSE dates back to the 19^{th} century. The establishment of the *International Settlement* in 1842 in Shanghai promoted the emergence of foreign markets and securities trading in the city. The first share list began to appear in June 1866, and in 1891 China's first stock exchange was established as the *Shanghai Stockbroker's Association*. In 1904, the organization was renamed to *Shanghai Stock Exchange*. In 1909, another Shanghai Broker's association was formed, and in 1920, two rival exchanges — *Shanghai Securities and Commodities Exchange* and *Shanghai Chinese Merchant Exchange* — were established. However, in 1929, an amalgamation occurred and the combined markets operated as the Shanghai Stock Exchange. Nevertheless, the SSE had to cease all operations in 1941 December when the Japanese military took over the International Settlement and Shanghai. The Shanghai stock market attempted to reestablish itself after the war but had to cease all operations again in 1949 during the communist revolution. It was only in 1990 that the present SSE opened [87, 88]. In less than a year after opening, the SSE adopted a centralized depository system for stocks, paving the way for paperless trading [89].

SSE equity market model

The SSE marketplace is made up of a trading system, a trading floor, Participant Business Unit, an order routing system and relevant communication systems [90]. Each trading day on the exchange is made up of three sub-periods: an opening call auction running from 09:15 to 09:25, and two continuous trading sessions, one from 09:30 to 11:30 and another from 13:00 to 15:00. Except for *B shares*, the exchange implements a designated trading system for all securities. The trading system accepts members' auction order routing during the opening call auction and both the continuous trading periods. Market and limit orders may be used to trade securities and the cancellation of orders is permitted only during continuous trading. Further, orders are matched by price-time priority during continuous trade. The tick size for the quotation price of *A shares* is RMB 0.01 Yuan and the maximum quantity for one order for a stock is one million shares. For a detailed outline on the market structure and rules of the SSE, refer to [90].

SSE50 index

The SSE portion of this research project concerned with stocks that constitute the SSE50 stock index. The SSE 50 is a stock index of SSE listed stocks, consisting of the 50 most liquid shares on SSE. For the selection criteria and calculation methodology of this index, refer to [91]. Appendix A.7 lists the constituents of the SSE 50 index from January 2010 to May 2016. Figure 3.55 illustrates the transactions prices and volumes of some SSE50 constituents on the 4^{th} of May 2016. Below, we consider the price response of trades for these stocks.

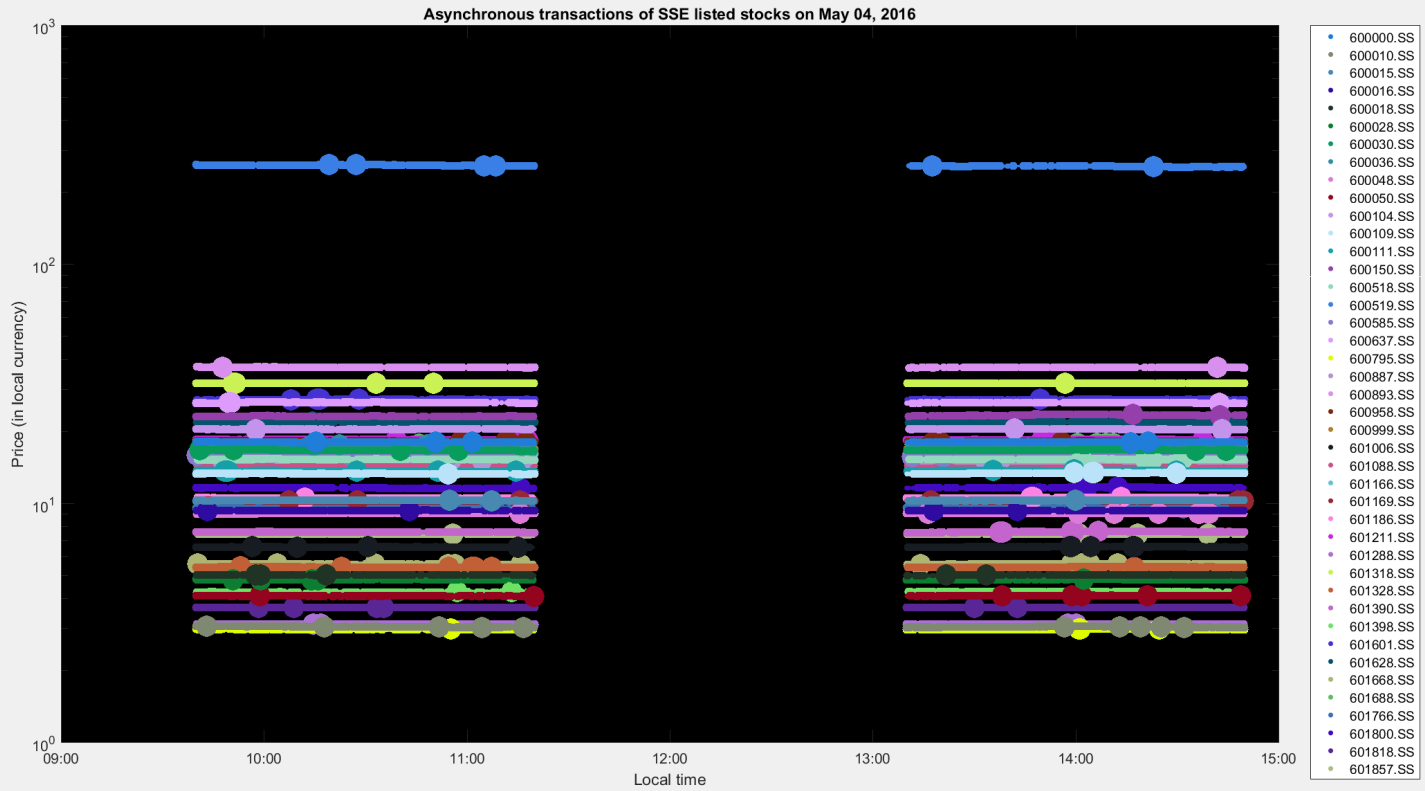


Figure 3.55: Transaction prices for SSE listed stocks on May 04, 2016. Each dot represents a transaction, and the size of the dot is proportional to the size of the transaction. The function provided in appendix C.7 was used to generate the scatter plot of each stock. The test script for this function is provided in appendix C.8.

Price impact for constituents of the SSE50

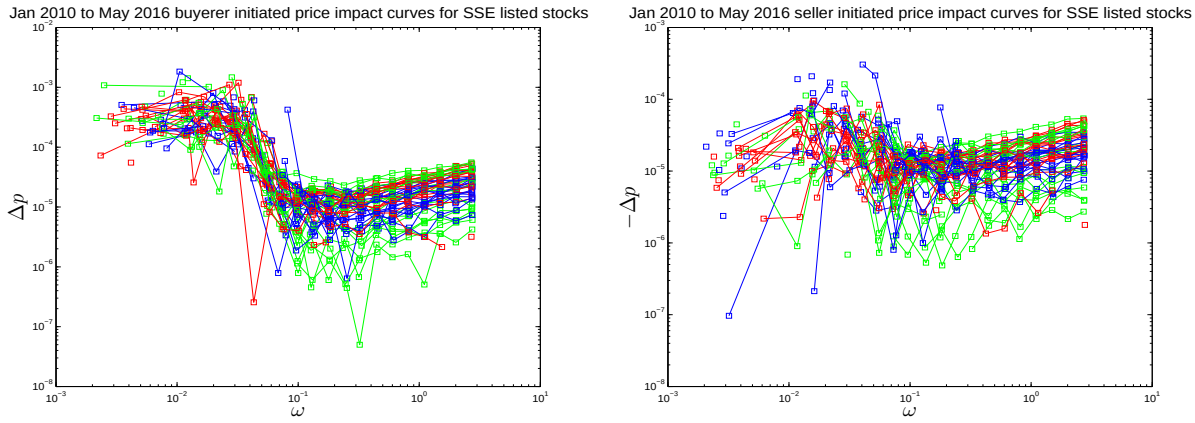


Figure 3.56: January 2010 to May 2016 price impact curves for constituents of the SSE50. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

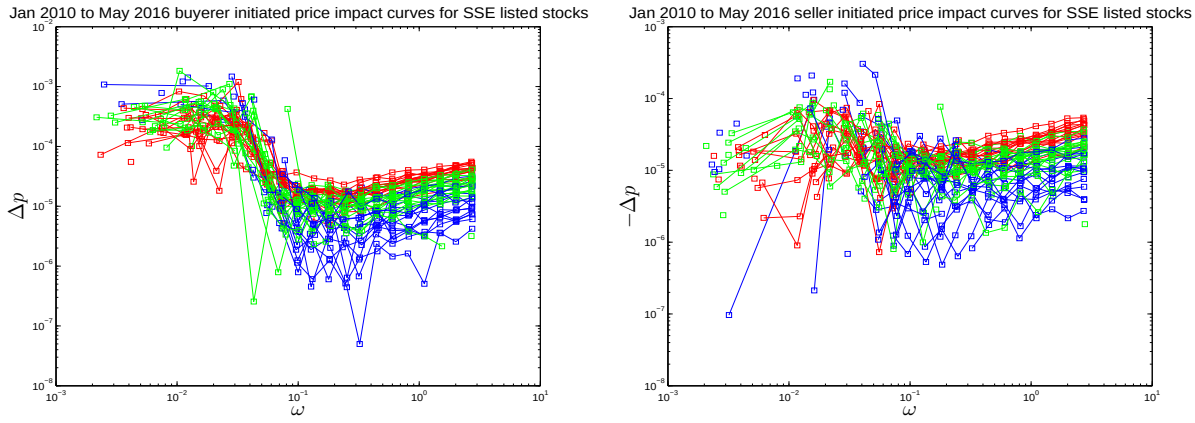


Figure 3.57: January 2010 to May 2016 price impact curves for constituents of the SSE50. Stocks were grouped into 3 different average daily volume (ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script to reproduce all of these figures can be found at https://github.com/Telmakaza.

Figure 3.56 illustrates the January 2010 to May 2016 average buyer and seller price impact for constituents of the SSE50. We note that all of the considered stocks have a price impact that initially decreases for small ω and then behaves like a power law for large values of ω. Turning our attention to figure 3.57, we note that there is a discernible relationship between ADV and average price impact for large values of ω: stocks that have a large ADV tend to induce a relatively small price impact Δp. However, from figure 3.56, we note that no discernible relationship between ADTV and price impact is clearly visible.

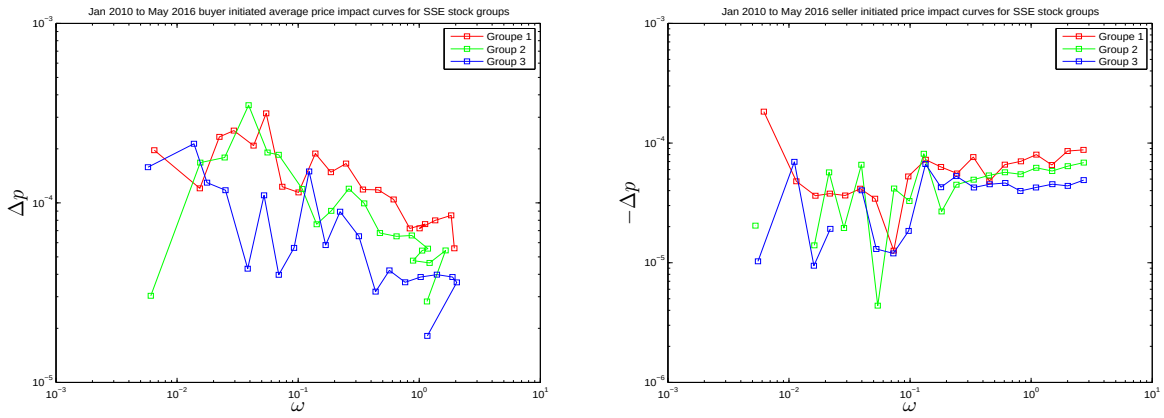


Figure 3.58: January 2010 to May 2016 average price impact curves for constituents of the SSE50. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. The average price impact was then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	α_b	λ_b	α_s	λ_s
Group 1	4.3548×10^9	3.6694×10^8	-0.2231	1.1369×10^4	0.0830	2.2059×10^4
Group 2	5.0066×10^9	5.0795×10^8	-0.2109	1.6343×10^4	0.1867	1.7789×10^4
Group 3	1.2523×10^{10}	1.0987×10^9	-0.2927	2.7229×10^4	0.2163	1.5475×10^4

Table 3.13: Calibrated price impact parameters for SSE stock groups.

In order to further investigate aggregate behaviour, using the method described in section 2.4.3 we constructed 3 stock groups, labelled *Group 1* to 3 respectively, and then calculated the average price impact for each stock group. The results that we obtained are shown in figure 3.58. The ADV and ADTV of each stock group is listed in table 3.13. From the plot on the left in figure 3.58, we note that the calculated buyer initiated price impacts seem to decrease for large ω. This contradicts the observations that we made in figure 3.58. However, we note that for both buyer and seller initiated trades, the price impacts of the stock groups arrange themselves from bottom to top in increasing order of ADTV and ADV. This finding is consistent with the observations made in the previously considered markets.

Trade sign autocorrelation and price dispersion

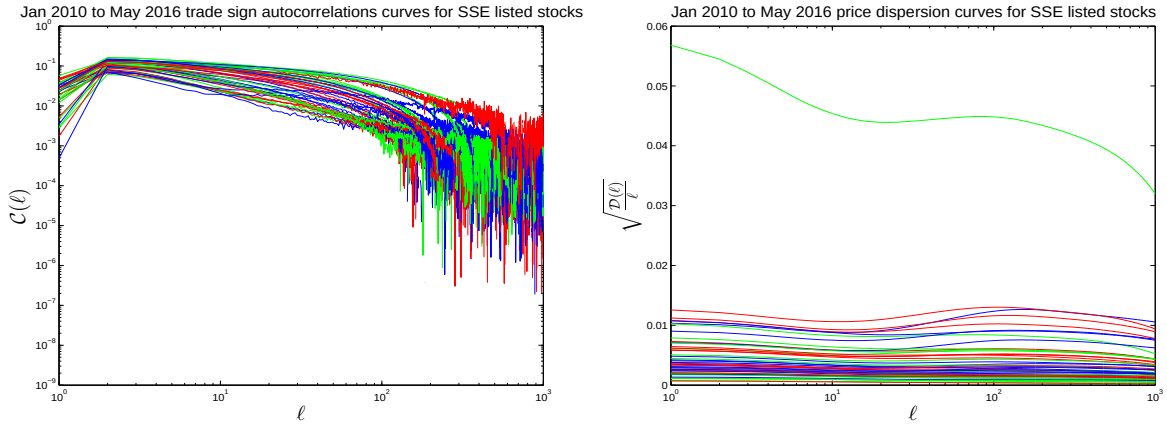


Figure 3.59: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the SSE50. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

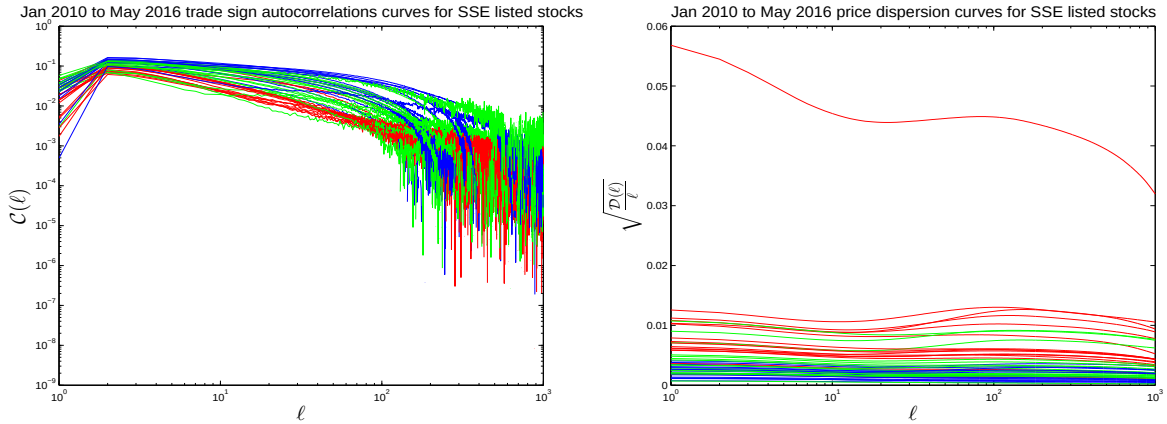


Figure 3.60: January 2010 to May 2016 average trade sign autocorrelation curves $\mathcal{C}(\ell)$ (left) and price dispersion curves (right) $\sqrt{\mathcal{D}(\ell)/\ell}$ for constituents of the SSE50. Stocks were grouped into 3 different average daily volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/.

Figure 3.59 visualises the January 2010 to May 2016 daily average trade sign autocorrelation function function $\mathcal{C}(\ell)$ and daily average normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$. We note that most of the considered stocks have a highly persistent $\mathcal{C}(\ell)$, and a $\sqrt{\mathcal{D}(\ell)/\ell}$ that exhibits sub-diffusion. Furthermore, unlike with the most of the previously considered markets, we note that it is difficult to establish a discernible relationship between

ADTV/ADV and $\mathcal{C}(\ell)$. However, from figure 3.60 we do note that stocks that have a relatively high ADV tend to have a relatively small normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$.

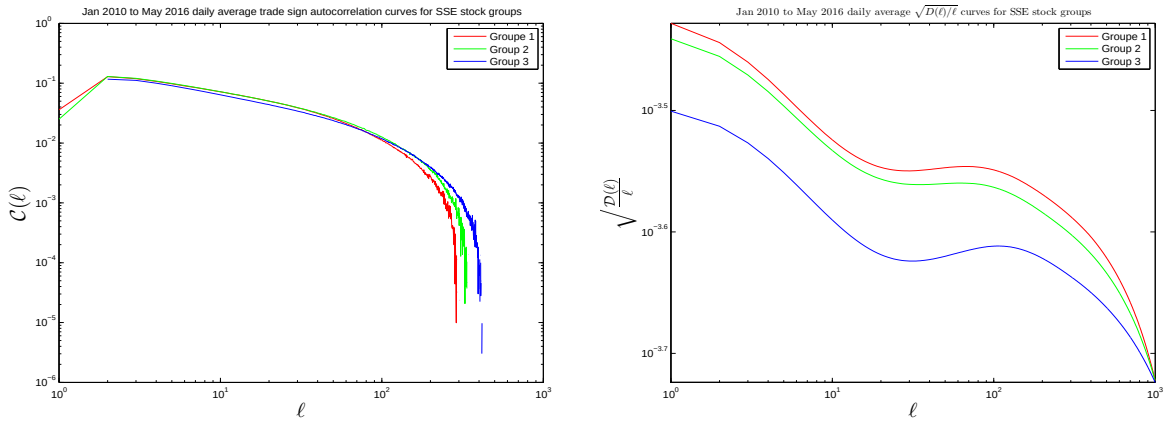


Figure 3.61: January 2010 to May 2016 daily average trade autocorrelation curves $\mathcal{C}(\ell)$ (left) and daily average normalised price dispersion curves $\sqrt{\mathcal{D}(\ell)/\ell}$ (right) for constituents of the SSE50. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	$\mathcal{C}_0$	γ
Group 1	4.3548×10^9	3.6694×10^8	0.1688	0.6076
Group 2	5.0066×10^9	5.0795×10^8	0.1658	0.5938
Group 3	1.2523×10^{10}	1.0987×10^9	0.0158	3.0929

Table 3.14: Calibrated $\mathcal{C}(\ell)$ parameters for SSE stocks groups

In order to further investigate the behaviour of $\mathcal{C}(\ell)$ and $\sqrt{\mathcal{D}(\ell)/\ell}$ for groups of stocks, we calculated these quantities for stock groups 1 to 3. The results that we obtained are shown in figure 3.61. We note that for large ℓ, a clear relationship emerges between ADV/ADTV and $\mathcal{C}(\ell)$: that that have a large ADV/ADTV tend to exhibit a high degree of trade sign autocorrelation $\mathcal{C}(\ell)$.

After fitting a relationship of the form

$$\mathcal{C}(\ell) = C_0 \ell^{\gamma} \tag{3.11}$$

to the calculated $\mathcal{C}(\ell)$, we obtained the parameters listed in table 3.14. We note that for Group 1 and 2, $0 < \gamma < 1$. This suggest that the trade sign autocorrelation process $\mathcal{C}(\ell)$ for these stock groups is a long memory process. However, we note that the normalised price dispersion process $\sqrt{\mathcal{D}(\ell)/\ell}$ of each stock group exhibits sub-diffusion. This may implies that each stock group has price changes that are not correlated. In the framework of [29], this may suggest that all of the considered stocks have a bare response function that decreases like a power law. In order to see if this is indeed the case, we investigate the price response coefficient and bare response function of all of these stocks in the next section.

Price response coefficient and bare response

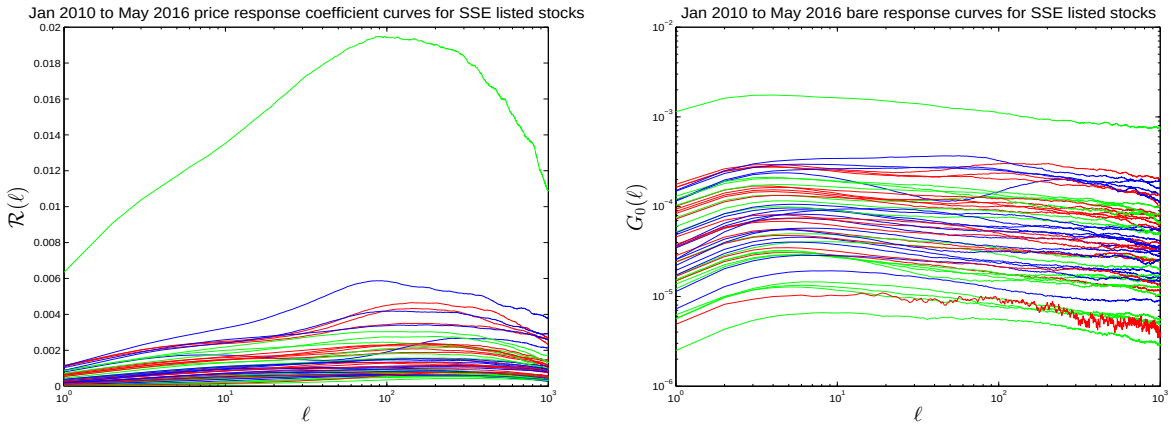


Figure 3.62: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the SSE50. Stocks were grouped into 3 different average daily traded value (ADTV) groups. Plots in red represent stocks in the low ADTV group, and those in green (respectively blue) represent stocks in the medium (high) ADTV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

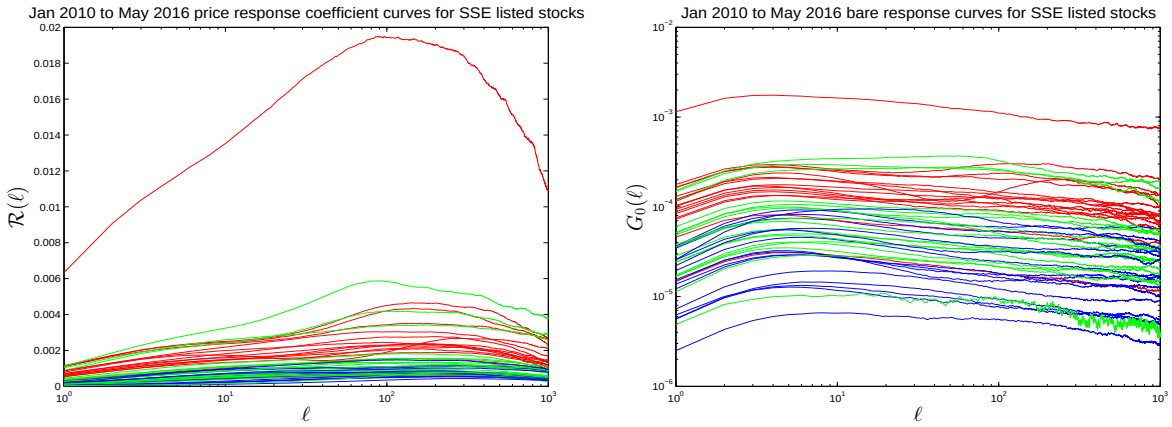


Figure 3.63: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ curves (left) and daily average bare response curves $G_0(\ell)$ for constituents of the SSE50. Stocks were grouped into 3 different average daily traded volume(ADV) groups. Plots in red represent stocks in the low ADV group, and those in green (respectively blue) represent stocks in the medium (high) ADV group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Figure 3.62 visualises the January 2010 to May 2016 daily average price response coefficient $\mathcal{R}(\ell)$ and daily average bare response function $G_0(\ell)$ for constituents of the SSE50. We note that all of the considered stocks have a $\mathcal{R}(\ell)$ initially increases for small values of

ℓ and then decreases for large ℓ. We also note that these stocks have a $G_0(\ell)$ that appears to first increase with ℓ and then decrease for larger values of ℓ.

Turning our attention to figure 3.63, we note that stocks that have a relatively high ADV tend to experience a relatively low price response, where price response is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$.

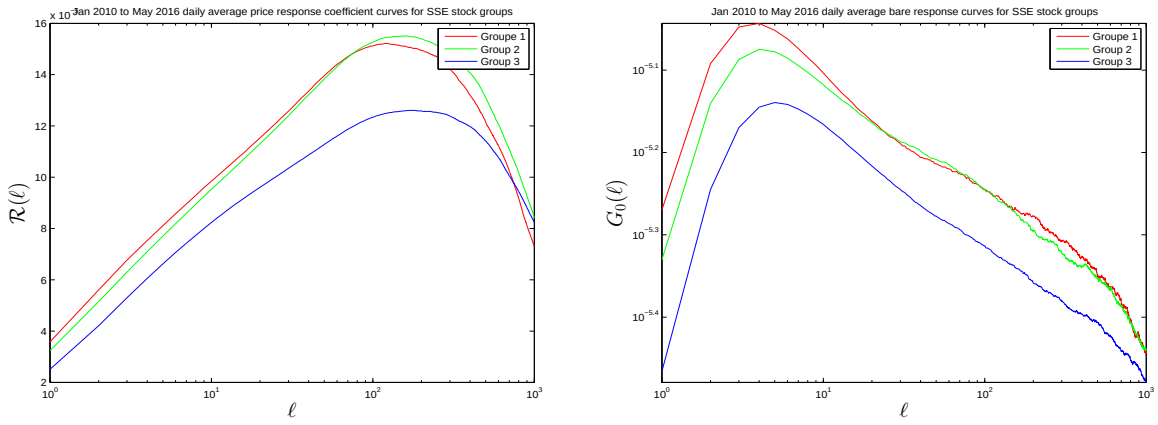


Figure 3.64: January 2010 to May 2016 daily average price response coefficient curves $\mathcal{R}(\ell)$ (left) and daily average bare response curves $G_0(\ell)$ (right) for constituents of the SSE. Stocks were grouped in such a way that the number of transactions in each group is roughly the same. $\mathcal{R}(\ell)$ and $G_0(\ell)$ were then calculated for each group. The underlying data and MATLAB script required to reproduce all of these figures are provided in the GitHub repository https://github.com/Telmakaza/Calculate_PriceResponse.

Stock group	ADVT	ADV	Γ_0	ℓ_0	β
Group 1	4.3548×10^9	3.6694×10^8	1.0563×10^{-5}	0	0.1402
Group 2	5.0066×10^9	5.0795×10^8	1.0247×10^{-5}	0	0.1373
Group 3	1.2523×10^{10}	1.0987×10^9	8.4567×10^{-6}	0	0.1264

Table 3.15: Calibrated $G_0(\ell)$ parameters for SSE stocks groups

Figure 3.64 visualises the calculated $\mathcal{R}(\ell)$ and $G_0(\ell)$ for stock groups 1 to three. We note that stock groups that have a relatively high ADTV and ADV tend to experience a relatively small price response, where price response is measured by $\mathcal{R}(\ell)$ and G_0. These findings are consistent with the observations that we made in most of the previously considered markets.

3.8 Summary of results (for all markets)

This section summaries and compares the results obtained for the considered markets. We consider the measured price impacts $\mathcal{R}(\ell = 1, V)$, trade sign autocorrelations $\mathcal{C}(\ell)$, price dispersions $\mathcal{D}(\ell)$, price response coefficients $\mathcal{R}(\ell)$ and bare response functions $G_0(\ell)$ respectively. Furthermore, we investigate the relationship between these measured quantities and liquidity, where liquidity is measured by average daily volume (ADV) and average daily traded value (ADTV). For a more detailed discussion of the below results, please refer to sections 3.1, 3.2, 3.3, 3.4, 3.5, 3.6 and 3.7.

3.8.1 Price Impact

For all of the considered markets, we found that the more actively traded stocks (i.e those with a relatively high ADV, ADTV and relatively large number of trades per day) tend to experience a lower degree of price impact, especially for large values of ω. Table 3.16 illustrates the calibrated price impact parameters for the B3, BSE, JSE, MOEX and SSE. Here, a relationship of the form

$$\ln \Delta p = \alpha \ln \omega - \ln \lambda \tag{3.12}$$

was fit to the price impacts of stock groups, labelled Group 1 to 3 respectively, for each market. The groups where created in such a way that the number of trades in each groups is roughly the same. Below, α_b and λ_b represent the estimated parameters for buyer initiated price impacts, and α_s and λ_s represent those for seller initiated price impacts.

Stock group	ADVT	ADV	α_b	λ_b	α_s	λ_s
B3 Group 1	0.4012×10^9	1.6975×10^7	0.1672	8.1954×10^3	0.0191	5.9460×10^3
B3 Group 2	0.4110×10^9	2.6600×10^7	0.2277	9.3909×10^3	0.2286	6.2073×10^3
B3 Group 3	1.3493×10^9	7.5033×10^7	0.2868	1.6438×10^4	0.3182	1.0984×10^4
BSE Group 1	7.9492×10^8	1.3664×10^6	0.1634	6.2433×10^3	0.2057	4.9385×10^3
BSE Group 2	9.5174×10^8	1.5694×10^6	0.1724	7.4710×10^3	0.2197	5.6609×10^3
BSE Group 3	1.7878×10^9	2.8197×10^6	0.1960	7.7118×10^3	0.2457	5.8723×10^3
JSE Group 1	9.1877×10^{10}	1.9393×10^7	0.0974	8.7522×10^3	0.0931	6.4551×10^3
JSE Group 2	1.3580×10^{11}	1.3834×10^7	0.1276	9.6878×10^3	0.0866	7.5588×10^3
JSE Group 3	4.0624×10^{11}	2.4874×10^7	0.1673	9.9782×10^3	0.1487	8.2908×10^3
MOEX Group 1	3.1304×10^9	4.4597×10^9	0.0591	2.4957×10^4	0.0278	1.6538×10^4
MOEX Group 2	4.1752×10^9	6.6207×10^9	0.1421	3.7657×10^4	0.1243	2.7629×10^4
MOEX Group 3	1.2043×10^{10}	1.3051×10^{10}	0.0955	5.5040×10^4	0.1766	3.1792×10^4
SSE Group 1	4.3548×10^9	3.6694×10^8	-0.2231	1.1369×10^4	0.0830	2.2059×10^4
SSE Group 2	5.0066×10^9	5.0795×10^8	-0.2109	1.6343×10^4	0.1867	1.7789×10^4
SSE Group 3	1.2523×10^{10}	1.0987×10^9	-0.2927	2.7229×10^4	0.2163	1.5475×10^4

Table 3.16: Calibrated price impact parameters for some of the considered markets.

From table 3.16, we can note that for the first four stock exchanges (i.e B3, BSE, JSE and MOEX), the behaviour of price impact is consistent. That is, more liquid stocks tend to have a lower but steeper price impact curve. This is true for both buyer and seller

initiated trades. However, the SSE has price impacts that do not confirm with the rest. In particular, we see that although buyer initiated trades tend to have price impacts that decrease with liquidity, the curves have a negative slope. Further, seller initiated price impacts decrease with liquidity but have positive slopes.

3.8.2 Trade sign autocorrelation and price dispersion

Turning our attention to the measured trade sign autocorrelation functions $\mathcal{C}(\ell)$ and normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$, we note that for the B3 and MOEX, the curves for $\mathcal{C}(\ell)$ tend to stack themselves, from top to bottom, in *decreasing* order of ADTV, ADV and average number of trades per day (see the plots on the left in figures 3.5, 3.7, 3.42, 3.44 and tables 3.2 and 3.11). This means that the more actively traded stocks in these markets tend to have a higher degree of trade sign autocorrelation. The curves for $\sqrt{\mathcal{D}(\ell)/\ell}$ are stacked, from top to bottom, in *increasing* order of ADV and number of trades per day (see the plots on the right in figures 3.6, 3.7, 3.43 and 3.44). This implies that the more actively traded stocks in these markets tend to have a smaller price dispersion.

For the BSE, stocks that have a relatively large ADV and ADTV tend have a larger degree of trade sign autocorrelation (see the plots on the right in figures 3.15 and 3.16). However, stocks that have a larger number of trades per day tend to have a smaller degree of $\mathcal{C}(\ell)$, especially for small values of ℓ (see the plot on the right in figure 3.17 and table 3.5). We note that BSE stocks with a relatively large ADV and number of trades per tend to have a smaller normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$ for all ℓ (see the plot on the right in figure 3.17 and table 3.5). However, we not that no clear relationship can be seen between ADTV and $\sqrt{\mathcal{D}(\ell)/\ell}$ for the considered BSE stocks (see the plot on the right in figure 3.15).

For both the EGX and NSE, we note that no clear relationship can be observed between ADTV and $\mathcal{C}(\ell)$ and ADTV and $\sqrt{\mathcal{D}(\ell)/\ell}$ (see figures 3.24 and 3.51). However, we note that the curves for $\mathcal{C}(\ell)$ are stacked from top to bottom in *decreasing* order of ADV, and those for $\sqrt{\mathcal{D}(\ell)/\ell}$ are arranged (from top to bottom) in *increasing* order of ADV (see figures 3.25 and 3.51).

For the JSE, we note that the stocks that have a relatively large number of trades per day tend to exhibit a smaller degree of trade sign autocorrelation $\mathcal{C}(\ell)$ (see figures 3.34 and table 3.8). This observation is similar to that made for the considered BSE stocks (see the plot on the left in figure 3.17) and contradicts that made in B3 and MOEX stocks (see figure 3.7 and 3.44). We also note that no clear relationship was observed between ADTV and $\mathcal{C}(\ell)$ (see the plot on the left in figure 3.32), between ADV and $\mathcal{C}(\ell)$ (see the plot on the left in figure 3.33) and between ADTV and $\sqrt{\mathcal{D}(\ell)/\ell}$ (see the plot on the right in figure 3.32) for the considered JSE stocks. However, we note the JSE stocks that have a relatively large ADV tend to have a smaller normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$ (see the plot on the right in figures 3.33 and 3.34 and table 3.8).

For the SSE, we note that no clear relationship was observed between $\mathcal{C}(\ell)$ and ADTV and $\sqrt{\mathcal{D}(\ell)/\ell}$ and ADTV (see the plot on the left in figure 3.59). However, we note that stocks that have a relatively large ADV tend to have a larger degree of trade sign autocorrelation $\mathcal{C}(\ell)$, and a smaller degree of normalised price dispersion $\sqrt{\mathcal{D}(\ell)/\ell}$ (see figure 3.60).

For a more quantitative illustration of some of these results, refer to table 3.17. Here, we fit a relationship of the form

$$\mathcal{C}(\ell) = C_0 \ell^{\gamma} \tag{3.13}$$

to the calculated $\mathcal{C}(\ell)$ for each stock group in the markets: B3, BSE, JSE, MOEX and SSE.

Stock group	ADVT	ADV	$\mathcal{C}_0$	γ
B3 Group 1	0.4012×10^9	1.6975×10^7	0.3144	0.5486
B3 Group 2	0.4110×10^9	2.6600×10^7	0.3473	0.4884
B3 Group 3	1.3493×10^9	7.5033×10^7	0.3588	0.4409
BSE Group 1	7.9492×10^8	1.3664×10^6	0.2049	0.6611
BSE Group 2	9.5174×10^8	1.5694×10^6	0.1672	0.6278
BSE Group 3	1.7878×10^9	2.8197×10^6	0.1165	0.5444
JSE Group 1	9.1877×10^{10}	1.9393×10^7	0.3514	0.6294
JSE Group 2	1.3580×10^{11}	1.3834×10^7	0.3362	0.6729
JSE Group 3	4.0624×10^{11}	2.4874×10^7	0.3112	0.6557
MOEX Group 1	3.1304×10^9	4.4597×10^9	0.3986	0.5705
MOEX Group 2	4.1752×10^9	6.6207×10^9	0.4291	0.5256
MOEX Group 3	1.2043×10^{10}	1.3051×10^{10}	0.3911	0.4736
SSE Group 1	4.3548×10^9	3.6694×10^8	0.1688	0.6076
SSE Group 2	5.0066×10^9	5.0795×10^8	0.1658	0.5938
SSE Group 3	1.2523×10^{10}	1.0987×10^9	0.0158	3.0929

Table 3.17: Calibrated $\mathcal{C}(\ell)$ parameters for B3, BSE, JSE, MOEX and SSE stocks groups

3.8.3 Price response coefficient and bare response

Turning our attention to the measured price response coefficient $\mathcal{R}(\ell)$ and bare response function $G_0(\ell)$, we note that for all of the considered markets, stocks that have a relatively high ADV and/or a relatively large number of trades per day tend to have a smaller degree of price response, where price response is measured by $\mathcal{R}(\ell)$ and $G_0(\ell)$ (see figures 3.9, 3.10, 3.19, 3.20, 3.27, 3.36, 3.37, 3.54, 3.63, 3.64 and table 3.18). However, from a visual inspection of the obtained results (see figures 3.8, 3.18, 3.26, 3.35, 3.53 and 3.62) no clear relation can be established between ADTV and $\mathcal{R}(\ell)$ and ADTV and $G_0(\ell)$. For a more quantitative illustration of the obtained results, please refer to table 3.18. Here, we fit a relationship of the form

$$G_0(\ell) = \frac{\Gamma_0}{(\ell_0 + \ell)^{\beta/2}} \tag{3.14}$$

to the measured $G_0(\ell)$ for each stock group, for the markets: B3, BSE, JSE, MOEX and SSE.

Stock group	ADVT	ADV	Γ_0	ℓ_0	β
B3 Group 1	0.4012×10^9	1.6975×10^7	5.5235×10^{-5}	0	0.4754
B3 Group 2	0.4110×10^9	2.6600×10^7	3.8795×10^{-5}	0	0.4909
B3 Group 3	1.3493×10^9	7.5033×10^7	2.2591×10^{-5}	0	0.4519
BSE Group 1	7.9492×10^8	1.3664×10^6	1.1538×10^{-5}	0	-0.8959
BSE Group 2	9.5174×10^8	1.5694×10^6	3.8681×10^{-5}	0	0.2051
BSE Group 3	1.7878×10^9	2.8197×10^6	3.0424×10^{-6}	0	0.2166
JSE Group 1	9.1877×10^{10}	1.9393×10^7	4.4306×10^{-5}	0	0.3875
JSE Group 2	1.3580×10^{11}	1.3834×10^7	4.7733×10^{-5}	0	0.3480
JSE Group 3	4.0624×10^{11}	2.4874×10^7	3.8792×10^{-5}	0	0.3480
MOEX Group 1	3.1304×10^9	4.4597×10^9	0	0	0
MOEX Group 2	4.1752×10^9	6.6207×10^9	1.2957×10^{-5}	0	0.3624
MOEX Group 3	1.2043×10^{10}	1.3051×10^{10}	6.2056×10^{-8}	0	-1.4911
SSE Group 1	4.3548×10^9	3.6694×10^8	1.0563×10^{-5}	0	0.1402
SSE Group 2	5.0066×10^9	5.0795×10^8	1.0247×10^{-5}	0	0.1373
SSE Group 3	1.2523×10^{10}	1.0987×10^9	8.4567×10^{-6}	0	0.1264

Table 3.18: Calibrated $G_0(\ell)$ parameters for B3, BSE, JSE, MOEX and stocks groups.

After consolidating the results in tables 3.16, 3.17 and 3.18 we can make the following conclusion for each stock exchange. The B3 and JSE have price impacts that first initially decrease and then increase like a power law for large trade volumes. This may imply that the rate of quote replenishment is relatively small for small trade volumes in these exchanges. BSE and MOEX on the other hand have price impacts that are roughly linear (on a double log plot) for all trade volumes. This may imply that these markets experience a more consistent quote replenishment pattern across all trade volumes. This holds for both buyer and seller initiated trades. The SSE deviates for the norm. It has buyer price impacts that are predominantly downwards sloping (although the curves do tend to slope upwards for very large trade volumes) and seller price impacts that slope upwards. This may suggest that the quote replenishment pattern that the market makers use in this market is vastly different from that used in the B3, JSE, BSE and MOEX. However, like

the other markets, the price impact curves of the SSE tend to decrease with liquidity for both buyer and seller initiated trades.

From table 3.17, we can see that the behaviour of trade sign autocorrelations varies across the different exchanges. BSE, JSE and SSE have trade sign autocorrelations that decrease with liquidity. This implies that in these markets, participants tend to increase the use of successive order splitting for less liquid stocks. The opposite behaviour is observed in the B3. The behaviour of $\mathcal{C}(\ell)$ is mixed in the MOEX market: stocks that have a "medium" liquidity (i.e stocks in *Group 2*) have the largest trade sign autocorrelation while stocks in the upper and lower liquidity range have a lower trade sign autocorrelation.

Lastly, from table 3.18, we we note that for B3 and the SSE, the bare response parameter Γ_0 decrease with an increase in both ADV and ADTV. Thus, both these measures of liquidity can be used as proxies for bare impact and therefore transaction cost in these markets. For the JSE, Γ_0 decrease with an increase in ADV. Thus, ADV can be used a transaction cost proxy here. For the BSE and MOEX, Γ_0 shows no clear relationship with either ADTV or ADV, so it is difficult to say if these measures of liquidity can proxy transaction costs in these markets.

Chapter 4

Conclusion

Tick data are financial markets data sets that record market price updates at every tick. These data sets are a significant resource because, among other things, they can be used to:

- Monitor the "health" today's, fast paced and ever changing markets [13].
- Drive data driven innovation and policy formation [14].
- Conduct data driven market microstructure and econophysics research [8, 15].
- Back test trading strategies and estimate pre- and post-trade execution costs [16].

However, tick data sets are often large in volume, inhomogeneous in structure and contaminated with erroneous data. Therefore, attempting to conduct tick data intensive research can prove to be a difficult task. Thus, the first objective of this dissertation was to implement and formalise a reproducible data processing *workflow*—similar to the one described in Hendricks *et al* (2016)[92]—to analyse relatively large volumes of tick data.

In particular, we considered a tick data set, from the Thompson Reuters Tick History database, for a selection of emerging markets traded stocks. The data spanned from January 2010 to May 2016 and the considered markets were *Brasil Bolsa Balcão* (B3), the *Bombay Stock Exchange* (BSE), the *Egyptian Exchange* (EGX), the *Johannesburg Stock Exchange* (JSE), the *Moscow Exchange* (MOEX), the *Nairobi Securities Exchange* (NSE) and the *Shanghai Stock Exchange* (SSE).

The workflow that we implemented for this data was made up of two components:

a) A database management system (DBMS) and

b) an application programming interface (API) that provides seamless access to the database.

We used a NoSQL database, MongoDB, as our database management system (DBMS). The main function of this database was to efficiently store the tick data and to enable the user to make data queries. The data accessed API, developed in MATLAB, served as an interface between the client and the database server: it enabled the client to interact with the database and store queried data to a MATLAB workspace for further processing. To enhance the reproducibility of this workflow, we provided a technical documentation of

its construction in appendix B.

Once the construction of the workflow was complete, we were able to carry out the second and main objective of this dissertation: to analyse and compare the *price response* of trades for a selection of emerging market traded stocks for the considered data set. Price response is the effect of a trade on subsequent prices: buyer initiated trades tend to push the price upwards and seller downwards. Some of the most important question concerning price response are related to temporal dependence (do prices stay the same or revert back to their average levels after "responding" to trades?) and volume dependence (do larger trades induce a more significant price response?). In order to try to answer questions relating to the temporal dependence of price response, we estimated the price response coefficient $\mathcal{R}(\ell)$, trade sign autocorrelation function $\mathcal{C}(\ell)$, normalised price dispersion $\mathcal{D}(\ell)$ and bare response function $G_0(\ell)$ of each consider stock; to answer questions related to the volume dependence of price response, we estimated the price impact function $\mathcal{R}(\ell = 1, V)$ for each stock. Here, the procedures described in Bouchaud *et al* (2003)[29] were used to calculate $\mathcal{R}(\ell)$, $\mathcal{C}(\ell)$, $\mathcal{D}(\ell)$ and $G_0(\ell)$. Procedures described in Lillo *et al* (2002)[32] and Lim and Coggins (2005)[35] were used to measure $\mathcal{R}(\ell = 1, V)$.

On the temporal dependence of price response, we found that for most of the considered stocks, the price response of trades tend to decrease with time (see figures 3.9, 3.10, 3.19, 3.20, 3.27, 3.36, 3.37, 3.54, 3.63, 3.64 and table 3.18). Further, we found that stocks that have a relatively large average daily volume tend to have a smaller degree of price response (see figures 3.10, 3.37, 3.20 and 3.64 and table 3.18). On the volume dependence of price response, we found that for most of the considered stocks, the price impact of trades tends to increase with the transacted volume (see section 3.8.1). Lastly, found that stocks that have a relatively large average daily volume or average daily value traded tend to experience a smaller degree of price impact (see section 3.8.1). Considering all of the above results, since both temporal and volume dependant price response are synonymous with transaction costs, one could argue that average daily volume could be used as a crude approximation for transaction costs.

In addition to this, we made interesting observations for each exchange. The price impact curves for the JSE, SSE and B3 stocks initially decrease and then increase for large trade volumes. This finding is consistent with observations made in M.Lim and Coggins (2005)[35] and Harvey *et al* (2017)[23]. As suggested by Harvey *et al* (2017)[23], this may be caused by an inconsistent level of order book resilience across trade volumes: smaller trade volumes may be subject to a degree level of quote replenishment and this permits for higher levels of price impact at low volumes in these markets. The BSE and MOEX on the other hand have price impact curves that are roughly linear (on a double log plot) across all trading volumes. This implies that the quote replenishment patterns in these exchanges are roughly consistent throughout all of the trading volumes, and this permits for a more *smoother* price impact. Lastly, we saw that the NSE and EGX have price impact curves that deviate from the norm: most of the curves form broken lines and don't seem to observe a power law of the form ω^{α}, even at large trade volumes. We postulate that this is because the level of liquidity in these markets is relatively sparse compared to the others.

Secondly, we found that the behaviour of trade sign autocorrelations varies across the different exchanges. BSE, JSE and SSE have trade sign autocorrelations that decrease with liquidity. This implies that in these markets, participants tend to increase the use of

successive order splitting for less liquid stocks. The opposite behaviour is observed in the B3. The behaviour of $\mathcal{C}(\ell)$ is mixed in the MOEX market: stocks that are in the medium liquidity range have the largest trade sign autocorrelation while stocks in the upper and lower liquidity range have a lower trade sign autocorrelation. Lastly, we note that for B3 and the SSE, the bare response parameter Γ_0 decrease with an increase in both ADV and ADTV. Thus, both these measures of liquidity can be used as proxies for bare impact and therefore transaction cost in these markets. For the JSE, Γ_0 decrease with an increase in ADV. Thus, ADV can be used a transaction cost proxy here. For the BSE and MOEX, Γ_0 shows no clear relationship with either ADTV or ADV, so it is difficult to say if these measures of liquidity can proxy transaction costs in these markets.

Price response and price impact have been widely studied for *develop* markets. For example, see Lim and Coggins (2005)[35], Lillo *et al* (2002)[32] and Bouchaud *et al*(2008)[55]. This study, along with Du Preez *et al* (2015)[22] and Harvey *et al* (2017)[23] bridge the gap in the literature by investigating price impact in *developing* markets. Our findings show that price response developing markets behaves in a similar manner to what has already been established in developed markets. In particular, if we use ADV as a proxy for market capitalisation, the price impact results that we obtained for BRICS markets (B3, MOEX, BSE, SSE and JSE) are similar to those obtained in Lillo *et al* (2002)[32] and Lim and Coggins (2005)[35]. That is, we found that the price impact curves are predominantly upward sloping and arrange themselves from top to bottom in an increasing order of some liquidity proxy, where we use ADV as the liquidity proxy and the latter use market capitalisation.

Similarly, Bouchaud *et al* (2003)[29] and Lillo and Farmer (2004) [40] independently reported highly persistent trade sign autocorrelation $\mathcal{C}(\ell)$ for some stocks listed on the Paris Stock Exchange and the London Stock Exchange. Bouchaud *et al* (2003)[29] argues that the autocorrelation persists (even after a large number of trades) because liquidity is thin in the market and trades split their orders into smaller pieces to avoid casing a large market impact. Our findings show that most of the considered stocks in the BRICS market also have a highly persistent trade sign autocorrelation, and by the argument raised by Bouchaud *et al* (2003)[29], market participants in these markets *too* behave like those in the Paris Stock Exchange and the London Stock Exchange by splitting large orders into smaller chunks to avoid excessive market impact. However, for EGX and NSE listed stocks, we found that trade sign autocorrelations are not as persistent as those for the considered BRICS listed stocks. This may suggest that there is a lower degree of successive order splitting in these markets.

As discussed in Lillo and Farmer (2004) [40], highly persistent trade sign autocorrelations are not consistent with uncorrelated price changes if price impact is permanent. Indeed, observersations of the diffusion function $\mathcal{D}(\ell)$ in Bouchaud *et al* (2003)[29] and in this dissertation show that price changes are predominantly uncorrelated. In order to reconcile these apparently contradictory observations (that is, high persistent trade sign autocorrelations and uncorrelated price changes), Bouchaud *et al* (2003)[29] proposed and defined a model under which the price impact at any point in time is a results of all past trades. Here, the price impact is mediated by the price response function $G_0(\ell)$ that describes the response to market of a single trade. Under this model, the author showed that in order for persistent trade sigh autocorrelation to be consistent uncorrelated price changes, the bare impact function $G_0(\ell)$ should decay like a power law suggesting that market makers mean revert the price levels to counteract the persistence of trade sign autocorrelations. Indeed Bouchaud *et al* (2008)[55] showed that this is the case for a

selection of four stocks. And this is precisely the result that we obtained for most of the considered stocks in this study: the calculated price response functions $\mathcal{R}(\ell)$ give evidence of mean reversion of the price level and the bare response functions $G_0(\ell)$ decay like a power law.

In conclusion our results show that market participants in developing markets behave in a similar way to those in developed markets, even though the countries are vastly different in terms of level of stock market liquidity and level of economic development. This may suggest that there is some kind of universal law governing the behaviour of market participants in order driven markets.

Appendices

Appendix A

Tables of considers stocks (by exchange)

A.1 .BSENSEX constituents

RIC	Company name	ICB
APSE.BO	Adani Port and Special Economic Zone Ltd	2000 Industrials
ASPN.BO	Asian Paints Ltd	1000 Basic Materials
AXBK.BO	Axis Bank Ltd	8000 Financials
BAJA.BO	Bajaj Auto Ltd	2000 Industrials
BHEL.BO	Bharat Heavy Electricals Ltd	2000 Industrials
BRTI.BO	Bharti Airtel Ltd	6000 Telecommunications
CIPL.BO	Cipla Ltd	4000 Health Care
COAL.BO	Coal India Ltd	1000 Basic Materials
GAIL.BO	GAIL (India) Ltd	7000 Utilities
HDBK.BO	HDFC Bank Ltd	8000 Financials
HDFC.BO	Housing Development Finance Corporation Ltd	8000 Financials
HLL.BO	Hindustan Unilever Ltd	3000 Consumer Goods
HROM.BO	Hero MotoCorp Ltd	2000 Industrials
ICBK.BO	ICICI Bank Ltd	8000 Financials
INFY.BO	Infosys Ltd	9000 Technology
ITC.BO	ITC Ltd	3000 Consumer Goods
LART.BO	Larsen & Toubro Ltd	2000 Industrials
LUPN.BO	Lupin Ltd	4000 Health Care
MAHM.BO	Mahindra and Mahindra Ltd	2000 Industrials
MRTI.BO	Maruti Suzuki India Ltd	2000 Industrials
NTPC.BO	NTPC Ltd	7000 Utilities
ONGC.BO	Oil and Natural Gas Corporation Ltd	0001 Oil & Gas
REDY.BO	Dr.Reddy's Laboratories Ltd	4000 Health Care
RELI.BO	Reliance Industries Ltd	0001 Oil & Gas
SBI.BO	State Bank of India	8000 Financials
SUN.BO	Sun Pharmaceutical Industries Ltd	4000 Health Care
TAMO.BO	Tata Motors Ltd	2000 Industrials
TCS.BO	Tata Consultancy Services Ltd	9000 Technology
TISC.BO	Tata Steel Ltd	1000 Basic Materials
WIPR.BO	Wipro Ltd	9000 Technology

Table A.1: List of considered stocks from the Bombay Exchange

A.2 Bovespa Index constituents

RIC	Company name	ICB
.BVSP4	No Information	No Information
ABEV3.SA	Ambev SA	3000 Consumer Goods
BBAS3.SA	Banco do Brasil SA	8000 Financials
BBDC3.SA	Banco Bradesco SA	8000 Financials
BBDC4.SA	Banco Bradesco SA	8000 Financials
BBSE3.SA	BB Seguridade Participacoes SA	8000 Financials
BRAP4.SA	Bradespar SA	1000 Basic Materials
BRFS3.SA	BRF SA	3000 Consumer Goods
BRKM5.SA	Braskem SA	1000 Basic Materials
BRML3.SA	BR Malls Participacoes SA	8000 Financials
BVMF3.SA	BM&F Bovespa SA Bolsa de Valores Mercadorias e Futuros	8000 Financials
CCRO3.SA	CCR SA	2000 Industrials
CESP6.SA	CESP Companhia Energetica de Sao Paulo	7000 Utilities
CIEL3.SA	Cielo SA	8000 Financials
CMIG4.SA	Companhia Energetica de Minas Gerais CEMIG	7000 Utilities
CPFE3.SA	CPFL Energia SA	7000 Utilities
CPLE6.SA	Companhia Paranaense de Energia	7000 Utilities
CSAN3.SA	Cosan SA Industria e Comercio	0001 Oil & Gas
CSNA3.SA	Companhia Siderurgica Nacional	1000 Basic Materials
CTIP3.SA	Cetip SA Mercados Organizados	8000 Financials
CYRE3.SA	Cyrela Brazil Realty SA Empreendimentos e Participacoes	3000 Consumer Goods
ECOR3.SA	Ecorodovias Infraestrutura e Logistica SA	2000 Industrials
EMBR3.SA	Embraer SA	2000 Industrials
ENBR3.SA	EDP Energias do Brasil SA	7000 Utilities
EQTL3.SA	Equatorial Energia SA	7000 Utilities
ESTC3.SA	Estacio Participacoes SA	5000 Consumer Services
FIBR3.SA	Fibria Celulose SA	1000 Basic Materials
GGBR4.SA	Gerdau SA	1000 Basic Materials
GOAU4.SA	Metalurgica Gerdau SA	1000 Basic Materials
HGTX3.SA	Cia Hering	3000 Consumer Goods
HYPE3.SA	Hypermarcas SA	4000 Health Care
ITSA4.SA	Itausa Investimentos Itau SA	8000 Financials

Table A.2: List of considered stocks from BM&F Bovespa

RIC	Company name	ICB
ITUB4.SA	Itau Unibanco Holding SA	8000 Financials
JBSS3.SA	Jbs SA	3000 Consumer Goods
KLBN11.SA	Klabin SA	1000 Basic Materials
KROT3.SA	Kroton Educacional SA	5000 Consumer Services
LAME4.SA	Lojas Americanas SA	5000 Consumer Services
LREN3.SA	Lojas Renner SA	3000 Consumer Goods
MRFG3.SA	Marfrig Global Foods SA	3000 Consumer Goods
MRVE3.SA	MRV Engenharia e Participacoes SA	8000 Financials
MULT3.SA	Multiplan Empreendimentos Imobiliarios SA	8000 Financials
NATU3.SA	Natura Cosmeticos SA	3000 Consumer Goods
OIBR3.SA	Oi SA	6000 Telecommunications
PCAR4.SA	Companhia Brasileira de Distribuicao	5000 Consumer Services
PETR3.SA	Petroleo Brasileiro SA Petrobras	0001 Oil & Gas
PETR4.SA	Petroleo Brasileiro SA Petrobras	0001 Oil & Gas
QUAL3.SA	Qualicorp SA	8000 Financials
RADL3.SA	Raia Drogasil SA	5000 Consumer Services
RENT3.SA	Localiza Rent a Car SA	2000 Industrials
RUMO3.SA	Rumo Logistica Operadora Multimodal SA	2000 Industrials
SANB11.SA	Banco Santander Brasil SA	8000 Financials
SBSP3.SA	Companhia de Saneamento Basico do Estado de Sao Paulo - SABESP	7000 Utilities
SMLE3.SA	Smiles SA	2000 Industrials
SUZB5.SA	Suzano Papel e Celulose SA	1000 Basic Materials
TBLE3.SA	No Information	No Information
TIMP3.SA	Tim Participacoes SA	6000 Telecommunications
UGPA3.SA	Ultrapar Participacoes SA	0001 Oil & Gas
USIM5.SA	Usinas Siderurgicas de Minas Gerais SA Usiminas	1000 Basic Materials
VALE3.SA	Vale SA	1000 Basic Materials
VALE5.SA	Vale SA	1000 Basic Materials
VIVT4.SA	Telefonica Brasil SA	6000 Telecommunications
WEGE3.SA	WEG SA	2000 Industrials

Table A.3: List of considered stocks from BM&F Bovespa (continued).

A.3 EGX30 constituents

RIC	Company name	ICB
ACGC.CA	Arabia Cotton Ginning Co SAE	3000 Consumer Goods
AIND.CA	Arabia Investments Development Financial Investments Holding Co SAE	2000 Industrials
AMER.CA	Amer Group Holding Co SAE	8000 Financials
AUTO.CA	GB Auto SAE	2000 Industrials
CCAP.CA	Qalaa Holdings SAE	8000 Financials
CIEB.CA	Credit Agricole Egypt SAE	8000 Financials
COMI.CA	Commercial International Bank Egypt SAE	8000 Financials
EAST.CA	Eastern Co SAE	3000 Consumer Goods
EFID.CA	Edita Food Industries SAE	3000 Consumer Goods
EGTS.CA	Egyptian Resorts Co SAE	8000 Financials
EMFD.CA	Emaar Misr for Development SAE	8000 Financials
ESRS.CA	Ezz Steel Co SAE	1000 Basic Materials
ETEL.CA	Telecom Egypt Co SAE	6000 Telecommunications
GTHE.CA	Global Telecom Holding SAE	6000 Telecommunications
HELI.CA	Heliopolis Co for Housing and Development SAE	3000 Consumer Goods
HRHO.CA	EFG Hermes Holdings SAE	8000 Financials
JUFO.CA	Juhayna Food Industries SAE	3000 Consumer Goods
MNHD.CA	Medinet Nasr for Housing and Development SAE	2000 Industrials
OCDI.CA	Sixth of October Development and Investment Co SAE	8000 Financials
ORAS.CA	Orascom Construction Ltd	2000 Industrials
ORWE.CA	Oriental Weavers Carpet Co SAE	3000 Consumer Goods
OTMT.CA	Orascom Telecom Media and Technology Holding SAE	6000 Telecommunications
PHDC.CA	Palm Hills Developments Company SAE	8000 Financials
PIOH.CA	Pioneers Holding Company for Financial Investments SAE	8000 Financials
PORT.CA	Porto Group Holding SAE	8000 Financials
SVCE.CA	South Valley Cement Co SAE	1000 Basic Materials
SWDY.CA	El Sewedy Electric Co SAE	2000 Industrials
TMGH.CA	Talaat Mostafa Group Holding Co SAE	8000 Financials
UASG.CA	United Arab Stevedoring Co SAE	2000 Industrials
UEGC.CA	El Saeed Contracting and Real Estate Investment Co SAE	2000 Industrials

Table A.4: List of considered stocks from the Egyptian Exchange

A.4 JSE Top 40 constituents

RIC	Company name	ICB
AGLJ.J	Anglo American PLC	1000 Basic Materials
AMSJ.J	Anglo American Platinum Ltd	1000 Basic Materials
ANGJ.J	AngloGold Ashanti Ltd	1000 Basic Materials
APNJ.J	Aspen Pharmacare Holdings Ltd	4000 Health Care
BATJ.J	Brait SE	8000 Financials
BGAJ.J	Barclays Africa Group Ltd	8000 Financials
BILJ.J	BHP Billiton PLC	1000 Basic Materials
BTIJ.J	British American Tobacco PLC	3000 Consumer Goods
BVTJ.J	Bidvest Group Ltd	2000 Industrials
CCOJ.J	Capital & Counties Properties PLC	8000 Financials
CFRJ.J	Compagnie Financiere Richemont SA	3000 Consumer Goods
CPIJ.J	Capitec Bank Holdings Ltd	8000 Financials
DSYJ.J	Discovery Ltd	8000 Financials
FFAJ.J	Fortress Income Fund Ltd	8000 Financials
FFBJ.J	Fortress Income Fund Ltd	8000 Financials
FSRJ.J	FirstRand Ltd	8000 Financials
GRTJ.J	Growthpoint Properties Ltd	8000 Financials
INLJ.J	Investec Ltd	8000 Financials
INPJ.J	Investec PLC	8000 Financials
ITUJ.J	Intu Properties PLC	8000 Financials
MEIJ.J	Mediclinic International PLC	4000 Health Care
MNDJ.J	Mondi Ltd	1000 Basic Materials
MNPJ.J	Mondi PLC	1000 Basic Materials
MRPJ.J	Mr Price Group Ltd	3000 Consumer Goods
MTNJ.J	MTN Group Ltd	6000 Telecommunications
NEDJ.J	Nedbank Group Ltd	8000 Financials
NPNJn.J	Naspers Ltd	9000 Technology
NTCJ.J	Netcare Ltd	4000 Health Care
OMLJ.J	Old Mutual PLC	8000 Financials
RDFJ.J	Redefine Properties Ltd	8000 Financials
REIJ.J	Reinet Investments SCA	8000 Financials
SLMJ.J	Sanlam Ltd	8000 Financials

Table A.5: List of considered stocks from the JSE

RIC	Company name	ICB
REMJ.J	Remgro Ltd	8000 Financials
RMHJ.J	RMB Holdings Ltd	8000 Financials
RMIJ.J	Rand Merchant Investment Holdings Ltd	8000 Financials
SABJ.J	No Information	No Information
SBKJ.J	Standard Bank Group Ltd	8000 Financials
SHPJ.J	Shoprite Holdings Ltd	5000 Consumer Services
SNHJ.J	Steinhoff International Holdings NV	5000 Consumer Services
SOLJ.J	Sasol Ltd	0001 Oil & Gas
TBSJ.J	Tiger Brands Ltd	3000 Consumer Goods
VODJ.J	Vodacom Group Ltd	6000 Telecommunications
WHLJ.J	Woolworths Holdings Ltd	5000 Consumer Services
SHPJ.J	Shoprite Holdings Ltd	5000 Consumer Services
SLMJ.J	Sanlam Ltd	8000 Financials
SNHJ.J	Steinhoff International Holdings NV	5000 Consumer Services
SOLJ.J	Sasol Ltd	0001 Oil & Gas
TBSJ.J	Tiger Brands Ltd	3000 Consumer Goods
VODJ.J	Vodacom Group Ltd	6000 Telecommunications
WHLJ.J	Woolworths Holdings Ltd	5000 Consumer Services

Table A.6: List of considered stocks from the JSE (continued)

A.5 IRTS constituents

RIC	Company name	ICB
AFKS.MM	AFK Sistema PAO	2000 Industrial
AFLT.MM	Aeroflot-Rossiyskiye Avialinii PAO	2000 Industrial
AKRN.MM	Akron PAO	1000 Basic Materials
ALRS.MM	AK Alrosa PAO	1000 Basic Materials
BANE.MM	ANK Bashneft' PAO	0001 Oil & Gas
BANEp.MM	ANK Bashneft' PAO	0001 Oil & Gas
BSPB.MM	Bank Sankt-Peterburg PAO	8000 Financials
CHMF.MM	Severstal' PAO	1000 Basic Materials
DIXY.MM	Dixy Group PAO	5000 Consumer Services
EONR_p.MM	No information	No Information
FEES.MM	FSK YeES PAO	7000 Utilities
GAZP.MM	Gazprom PAO	0001 Oil & Gas
GCHE.MM	Gruppa Cherkizovo PAO	3000 Consumer Goods
GMKN.MM	GMK Noril'skiy Nikel' PAO	1000 Basic Materials
HYDR.MM	Federal Hydro-Generating Company RusHydro PAO	7000 Utilities
IRAO.MM	Inter RAO YEES PAO	7000 Utilities
LKOH.MM	NK Lukoil PAO	0001 Oil & Gas
LSRG.MM	Gruppa LSR PAO	8000 Financials
MAGN.MM	Magnitogorskiy metallurgicheskiy kombinat OAO	1000 Basic Materials
MFON.MM	MegaFon PAO	6000 Telecommunications
MGNT.MM	Magnit PAO	5000 Consumer Services
MOEX.MM	Moskovskaya Birzha MMVB-RTS PAO	8000 Financials
MSRS.MM	MOESK PAO	7000 Utilities
MSTT.MM	Mostotrest PAO	2000 Industrial
MTLR.MM	Mechel PAO	1000 Basic Materials
MTSS.MM	Mobil'nye Telesistemy PAO	6000 Telecommunications
MVID.MM	M.video PAO	3000 Consumer Goods
NLMK.MM	Novolipetsk Steel PAO	1000 Basic Materials
NMTP.MM	Novorossiyskiy Morskoy Torgovyi Port PAO	2000 Industrial
NVTK.MM	Novatek PAO	0001 Oil & Gas

Table A.7: List of considered stocks from the MOEX

RIC	Company name	ICB
PGIL.MM	No information	No information
PHOR.MM	PhosAgro PAO	1000 Basic Materials
PHST.MM	No information	No Information
PIKK.MM	Gruppa Kompaniy PIK PAO	3000 Consumer Goods
POLY.MM	Polymetal International PLC	1000 Basic Materials
ROSN.MM	NK Rosneft' PAO	0001 Oil & Gas
RSTI.MM	Rossiyskiye Seti PAO	7000 Utilities
RTKM.MM	Rostelekom PAO	6000 Telecommunications
RTKM_p.MM	Rostelekom PAO	6000 Telecommunications
RUALR.MM	No information	No information
SBER.MM	Sberbank Rossii PAO	8000 Financials
SBER.MM	Sberbank Rossii PAO	8000 Financials
SNGS.MM	Surgutneftegaz OAO	0001 Oil & Gas
SNGS.MM	Surgutneftegaz OAO	0001 Oil & Gas
SVAV.MM	Sollers PAO	2000 Industrials
TATN.MM	Tatneft' PAO	0001 Oil & Gas
TATNp.MM	Tatneft' PAO	0001 Oil & Gas
TRMK.MM	Trubnaya Metallurgicheskaya Kompaniya PAO	1000 Basic Materials
TRNF_p.MM	Transneft' PAO	0001 Oil & Gas
URKA.MM	Uralkaliy PAO	1000 Basic Materials
VSMO.MM	Korporatsiya VSMPO-AVISMA PAO	1000 Basic Materials
VTBR.MM	Bank VTB PAO	8000 Financials
YNDX.MM	Yandex NV	9000 Technology

Table A.8: List of considered stocks from the MOEX (continued)

A.6 NSE20 constituents

RIC	Company name	ICB
ARM.NR	ARM Cement Ltd	1000 Basic Materials
BAMB.NR	Bamburi Cement Ltd	1000 Basic Materials
BAT.NR	British American Tobacco Kenya Ltd	3000 Consumer Goods
BBK.NR	Barclays Bank of Kenya Ltd	8000 Financials
BRIT.NR	Britam Holdings Ltd	8000 Financials
CFC.NR	CfC Stanbic Holdings Ltd	8000 Financials
COOP.NR	Co-operative Bank of Kenya Ltd	8000 Financials
EABL.NR	East African Breweries Ltd	3000 Consumer Goods
EQTY.NR	Equity Group Holdings Ltd	8000 Financials
ICDC.NR	Centum Investment Company Ltd	8000 Financials
KCB.NR	KCB Group Plc	8000 Financials
KEGN.NR	Kenya Electricity Generating Co Ltd	7000 Utilities
KENO.NR	KenolKobil Ltd	0001 Oil & Gas
KPLC.NR	Kenya Power and Lighting Company Ltd	7000 Utilities
KQNA.NR	Kenya Airways Ltd	2000 Industrials
NMG.NR	Nation Media Group	5000 Consumer Services
SASN.NR	Sasini Ltd	3000 Consumer Goods
SCAN.NR	WPP-Scangroup Ltd	5000 Consumer Services
SCBK.NR	Standard Chartered Bank Kenya Ltd	8000 Financials
SCOM.NR	Safaricom Ltd	6000 Telecommunications

Table A.9: List of considered stocks from the NSE

A.7 SSE50 constituents

RIC	Company name	ICB
600000.SS	Shanghai Pudong Development Bank Co Ltd	8000 Financials
600010.SS	Inner Mongolia BaoTou Steel Union Co Ltd	1000 Basic Materials
600015.SS	Hua Xia Bank Co Ltd	8000 Financials
600016.SS	China Minsheng Banking Corp Ltd	8000 Financials
600018.SS	Shanghai International Port Group Co Ltd	2000 Industrials
600028.SS	China Petroleum & Chemical Corp	0001 Oil & Gas
600030.SS	CITIC Securities Co Ltd	8000 Financials
600036.SS	China Merchants Bank Co Ltd	8000 Financials
600048.SS	Poly Real Estate Group Co Ltd	8000 Financials
600050.SS	China United Network Communications Ltd	6000 Telecommunications
600104.SS	SAIC Motor Corp Ltd	2000 Industrials
600109.SS	Sinolink Securities Co Ltd	8000 Financials
600111.SS	China Northern Rare Earth Group High-Tech Co Ltd	1000 Basic Materials
600150.SS	China CSSC Holdings Ltd	2000 Industrials
600518.SS	Kangmei Pharmaceutical Co Ltd	4000 Health Care
600519.SS	Kweichow Moutai Co Ltd	3000 Consumer Goods
600585.SS	Anhui Conch Cement Co Ltd	1000 Basic Materials
600637.SS	Shanghai Oriental Pearl Media Co Ltd	5000 Consumer Services
600795.SS	GD Power Development Co Ltd	7000 Utilities
600837.SS	Haitong Securities Co Ltd	8000 Financials
600887.SS	Inner Mongolia Yili Industrial Group Co Ltd	3000 Consumer Goods
600893.SS	AVIC Aviation Engine Corp PLC	3000 Consumer Goods
600958.SS	Orient Securities Co Ltd	8000 Financials
600999.SS	China Merchants Securities Co Ltd	8000 Financials
601006.SS	Daqin Railway Co Ltd	2000 Industrials
601088.SS	China Shenhua Energy Co Ltd	1000 Basic Materials
601166.SS	Industrial Bank Co Ltd	8000 Financials
601169.SS	Bank of Beijing Co Ltd	8000 Financials
601186.SS	China Railway Construction Corp Ltd	2000 Industrials
601211.SS	Guotai Junan Securities Co Ltd	8000 Financials
601288.SS	Agricultural Bank of China Ltd	8000 Financials
601318.SS	Ping An Insurance Group Co of China Ltd	8000 Financials
601328.SS	Bank of Communications Co Ltd	8000 Financials
601336.SS	New China Life Insurance Co Ltd	8000 Financials
601390.SS	China Railway Group Ltd	2000 Industrials

Table A.10: List of considered stocks from the SSE

RIC	Company name	ICB
601398.SS	Industrial and Commercial Bank of China Ltd	8000 Financials
601601.SS	China Pacific Insurance Group Co Ltd	8000 Financials
601628.SS	China Life Insurance Co Ltd	8000 Financials
601668.SS	China State Construction Engineering Corp Ltd	2000 Industrials
601669.SS	Power Construction Corp of China Ltd	2000 Industrials
601688.SS	Huatai Securities Co Ltd	8000 Financials
601766.SS	CRRC Corp Ltd	2000 Industrials
601800.SS	China Communications Construction Co Ltd	2000 Industrials
601818.SS	China Everbright Bank Co Ltd	8000 Financials
601857.SS	PetroChina Co Ltd	0001 Oil & Gas
601901.SS	Founder Securities Co Ltd	8000 Financials
601985.SS	China National Nuclear Power Co Ltd	7000 Utilities
601988.SS	Bank of China Ltd	8000 Financials
601989.SS	China Shipbuilding Industry Co Ltd	2000 Industrials
601998.SS	China Citic Bank Corp Ltd	8000 Financials

Table A.11: List of considered stocks from the SSE

Appendix B

BRICSData database technical documentation

This appendix highlights the procedure that we followed to construct the *BRICSData* database. It also provides a "how to get started" guide for implementing a tick database similar to *BRICSData*. The layout is as follows. Section B.1 describes the method that we used to initialise *BRICSData* and section B.2 sketches the used data importation procedure. Sections B.3 and B.4 respectively discuss creating indices and enabled accessing control for the database. Section B.5 discuses the method that we employed to interact with and query data from the database. Lastly section B.6 provides the "how to get started guide" for the database implementation software that we provide at https://github.com/Telmakaza.

B.1 Creating the BRICSData database

The BRICSData database was developed on a Windows machine with the specifications listed in appendix D.1 and the database ran on MongoDB version 3.0. The first step in building the database was to configuring a Windows service for an instance of MongoDB. The procedure that we used to do this doing this is as follows.

1. First, we created two directories: one to store the data and another to store a log file to the database. The following commands were executed in the command prompt to archive this.

```
mkdir G:\BRICS\data

mkdir G:\BRICS\log
```

   The folder `G:\BRICS\data` became the data storage folder, and `G:\BRICS\log` the log file folder.

2. We then used a text editor to create a `.cfg` file called `mongo`. The contents of the file are:

```
systemLog:
    destination: file
```

```
    path: g:\BRICS\log\mongod.log
storage:
    dbPath: g:\BRICS\data
```

Note that this file specifies the data directory, `G:\BRICS\data`, and the directory of `mongo.log`, a yet to be created log file in the log file directory `G:\BRICS\log`.

3. Next we ensured that `mongod.exe`, a MongoDB executable, is appended to the `Path` environment variable in Windows.

4. We then executed the following command in the command prompt:

```
mongod ––config G:\BRICS\mongod.cfg ––serviceName MongoDB_BRICS
––serviceDisplayName MongoDB_BRICS ––install
```

This command configured a service called `MongoDB_BRICS` for an instance of MongoDB.

5. Next we started the configured service by executing the command

```
net start MongoDB_BRICS
```

6. The final step was to initialize the BRICSData database. This involved two steps. The first was to execute the command illustrated below in the command prompt

```
mongo
```

The above command starts the *mongo shell* and runs the `MongoDB_BRICS` instance of MongoDB. The last of the two steps of to create a new database called *BRICSData* by executing the command illustrated below in the mongo shell

```
use BRICSData
```

All that remained was to import the tick data into the database and to develop a means to query the database. The details of how this was done are discussed in the next sections.

B.2 Importing data into the database

We used `mongoinport`, a MongoDB executable, to import the data into the database. We created 7 *collections*, one for each considered market, inside the BRICSData database. The naming conversion for the collections is

`<Market>Transactions.`

Thus, the collection that contained the JSE data was named `JSETransactions`, and that which contained the BSE data was named `BSETransactions`, etc. We executed a commands of the form:

```
mongoimport –d <database> –c <collection> ––type csv ––file <filename>
––ignoreBlanks  ––fields "<Header>"
```

To upload the data to the database. Here, expressions enclosed inside angle braces do not represent literal expressions. `<database>` is the name of the destination database. In our case, this is always `BRICSData`. `<collection>` depends on the market data in question (that is, we chose it to be `JSETransactions` for JSE and `NSETransactions` for NSE data etc). `<filename>` represnts the file name of each csv file, and `<Header>` its header. In addition to this, the import commands included the option `--ignoreBlanks` . This option enables us to ignore columns that don't have any values in them. Appendix C.1 provides the precise data importation script that we used. Furthermore, we provide a worked example below to illustrate how this script works.

Data import worked example

Suppose that we have two csv files with the contents indicated in tables B.1 and B.2. The file name of each csv file is indicated in the caption below its table. Suppose that we would like to import these data files into a MongoDB collection named `JSETransactions`, under the `BRICSData` database (i.e the values of `<database>` and `<collection>` are `BRICSData` and `JSETransactions` respectively).

#RIC	GMT Offset	Date[G]	Time[G]	Type	Price	Volume	Bid Price	Ask Price
XYZJ.J	+2	04-JAN-2010	09:30:36.703	Trade	6394	36		
XYZJ.J	+2	04-JAN-2010	09:34:36.366	Quote			6583	6593

Table B.1: Hypothetical csv file with file name `JSEpart01.csv`

#RIC	GMT Offset	Date[G]	Time[G]	Type	Price	Volume	Ask Price	Bid Price
ABCJ.J	+2	04-JAN-2010	09:35:36.703	Quote			891	893
ABCJ.J	+2	04-JAN-2010	10:33:36.703	Quote			890	892

Table B.2: Hypothetical CSV file with file name `JSEpart02.csv`

Since these files are named `JSEpart01.csv` and `JSEpart02.csv` respectively, the `<filename>` input takes on a value of either `JSEpart01.csv` or `JSEpart02.csv`. An iterative procedure can be used to uploads these scripts into the database. This is precisely what our data importation script in appendix C.1 does. It constructs the data importation

execution sting for each file, and then executes the constructed execution string using the MATLAB `dos` function. In this example, the constructed execution strings would be

```
mongoimport –d BRICSData –c JSETransactions ––type csv ––file JSEpart01.csv
––ignoreBlanks  ––fields "#RIC,Date[G],Time[G],Type,Price,Volume,
Bid Prce,Ask Price"
```

and for the file `JSEpart01.csv`, and

```
mongoimport –d BRICSData –c JSETransactions ––type csv ––file JSEpart02.csv
––ignoreBlanks  ––fields "#RIC,Date[G],Time[G],Type,Price,Volume,
Bid Prce,Ask Price"
```

for `JSEpart02.csv`. Figure B.1 illustrates what the data would look like once inside the database.

```
{
      "_id":ObjectId(3sds644bbea80b01e4d75e)
      #RIC":XYZJ.J
      "Date[G]":"04-JAN-2010"
     "Time[G]":"09:30:36.703"
      "Type":"Trade"
      "Price":  6394
      "Volume":  36

}
{
      "_id":ObjectId(58f61f4bbea8dfs0gg69df)
      #RIC":  XYZJ.J
      "Date[G]":"2010-01-04TZ"
      "Time[G]":  "09:34:36.366"
      "Type":"Quote"
      "Bid Price":  6583
      "Ask Price":  6593

}
{
      "_id":ObjectId(53ssdryth4bbea80b01e4d75e)
      #RIC":ABCJ.J
      "Date[G]":"04-JAN-2010"
      "Time[G]":"09:35:36.703"
      "Type":"Quote"
      "Bid Price":  891
      "Ask Price:" 893

}
{
      "_id":ObjectId(g8f61frggbej80b01e4dgr6)
      #RIC":ABCJ.J
      "Date[G]":"04-JAN-2010"
      "Time[G]":"10:33:36.703"
      "Type":"Quote"
      "Bid Price":  890
      "Ask Price:" 892

}
```


Figure B.1: Hypothetical documents in database (corresponding to hypothetical csv file in table 3) imported using the MATLAB script.

B.3 Building indexes

After importing all of the data into the database, we built compound indices for the values of the fields `#RIC` and `Date[G]`, for each collection. Doing this enabled us to query data by RIC and date at a relatively fast rate. The commands that we used to build the indices are of the form:

```
db<Market>Transactions.ensureIndexe({#RIC: 1, Date[G]: 1})
```

Here, `<Market>Transactions` represents the collection name. Thus, the exact command for that we used to build indices for the the `JSETransactions` collection is

```
db.JSETransactions.ensureIndex({#RIC: 1, Date[G]: 1})
```

and that for the `NSETransactions` collection is:

```
db.NSETransactions.ensureIndex({#RIC: 1, Date[G]: 1})
```

etcetera.

B.4 Enabling access control

After the data was uploaded to the database, we enabled user authentication to set up access control to the data tick set. In order to do this, we executed the following command in the mongo shell

```
db.createUser
(
    {
        user: "enonyane"
        pwd: "****"
        roles:
        [
            {role: "readWrite", db: "BRICSData"}
            {role: "dbAdmin", db: "BRICSData"}
        ]
    }
)
```

B.5 Querying data from the database

The MongoDB executable, `mongoexport`, was used as the main tool to conduct database queries. Database queries are commands of the form

```
mongoexport --host <host> --port <port> --authenticationDatabase <database>
 -u <username> -p <password> -d <database> -c <collection> -q "<query>"
 --type=csv --fieldFile <fieldfile> --out <output>
```

<host> is the I.P address of the database server, <port> is the port number, <database> is the name of the database that the user would like to query, <username> is the user name, <password> is the password used for access control, <collection> is the collection that the client would like to query, <query> is the database query, <fieldfile> specifies the path of a field file — a file that specifies the fields that the user would like to extract from the database. For this project, the values of these fields are as follows:

- <host> = 146.145.56.52
- <port> = 27017
- <username> = enonyane
- <collection> = <Market>Transations (e.g, we have <collection> = JSETransactions for the collection containing JSE data.)
- <query> = #RIC: <ric>, Date[G]: <date> . Here <ric> and <date> are user inputs (e.g we can have <ric> = BGAJ.J and <date> = 04-JAN-2010)
- <fieldfile> = csv file with the contents:

```
#RIC
Date[G]
Time[G]
Type
Price
Volume
Bid Price
Ask Price
```

Suppose that we would like to query the data of a stock with ticker #RIC on the date 4^{th} of January 2010, from a collection called JSETransactions, located in the BRICSData database. Further, suppose that we would like to extract the data — by the fields specified in a file called ff.csv, located in the directory C:\FieldFiles — to the output file C:\TempData\temp.csv. The command to do this is as follows:

```
mongoexport --host 146.145.56.52 --port 27017 --authenticationDatabase
BRICSData -u enonyane -p **** -d BRICSData -c JSETransaction
 -q "{ "#RIC" : "BGAJ.J", "Date[D]" : "04-JAN-2010"}"
 --type=csv --fieldFile C:\FieldFiles\ff.csv --out C:\TempData\temp.csv
```

This command would generate the csv file C:\TempData\temp.csv. The data in this file can then be imported directly into MATLAB and processed for analysis. The script provided in appendix C.2 implements the procedure described above.

B.6 Database implementation: example

This section presents a "how to get started" guide for the database implementation software that we provide at . The aim of this section is to enable the reader to replicate the main components of our tick data store *BRICSData*. The steps required to replicate the workflow are explained below. We note that the steps described here require the g++ complier. We used Cygwin, a large collection of GNU and Open Source tools, in order to access to g++ on a Windows machine[93].

Step 1: initialising the database

The first step is to initialise the database. This is relatively simple; it entails running the command

```
use BRICSData
```

in the mongo shell. Here `BRICSData` is the name of the database to be initialised.

Step 2: Creating directories for the raw data, job data and obtaining the source code

The next step is to create directories for the raw data, job data and source code. We chose our raw data directory to be `C:\TRTH\RawData` and the job data directory was `C:\TRTH\JobData`.

Step 3: Obtaining the source code and market data

The source code and a sample csv file of the market data that we used are provided in the GitHub repository https://github.com/Telmakaza/mongoDB_Tickstore. In order to implement our workflow, we recommend cloning this repository to you local machine.

Step 4: moving datafiles to the raw data directory and compiling the required code

Once the source code has been obtained, market data can be imported imported into the database initialised in *Step 1*. This requires one move (or copy) the provided market data to the raw data file directory created in *Step 2*. Secondly, one needs to compile a few .cpp files in the source folder in order to prepare the data importation process. This can be done by navigation to the code source folder in the provided repository and executing a command of the form:

```
g++ main.cpp getfilenames.cpp csv_tojson.cpp to_isodate.cpp –o <binDirectory>
```

on a the bash terminal (we used the Cygwin bash terminal). Here `g++ main.cpp getfilenames.cpp csv_tojson.cpp to_isodate.cpp -o` is a literal expression and `<binDirectory>/main` is the (machine dependent) directory of the `bin` folder in the provided repository. For our implantation, the directory of the repository was

C:\Users\ENonyane\Desktop, thus, the precise execution string required to compile the .cpp files was

```
g++ main.cpp getfilenames.cpp csv_tojson.cpp to_isodate.cpp -o
C:/Users/ENonyane/Desktop/mongoDB_Tickstore/bin/main
```

Here main.cpp defines a main C++ that calls getfilenames, a functions — defined in getfilenames.cpp — that gets the file names of all of the data files in the raw data directory. The function main function in main.cpp then calls toJSON, a function — defined in csv_tojson — that converts the data files in the raw data directory into .json files, which are stored in the job data directory defined in *Step 2*. The toJSON function requires toISO: a function defined in to_isodate.cpp that converts timestamps into an *International Organization for Standardization*-like format.

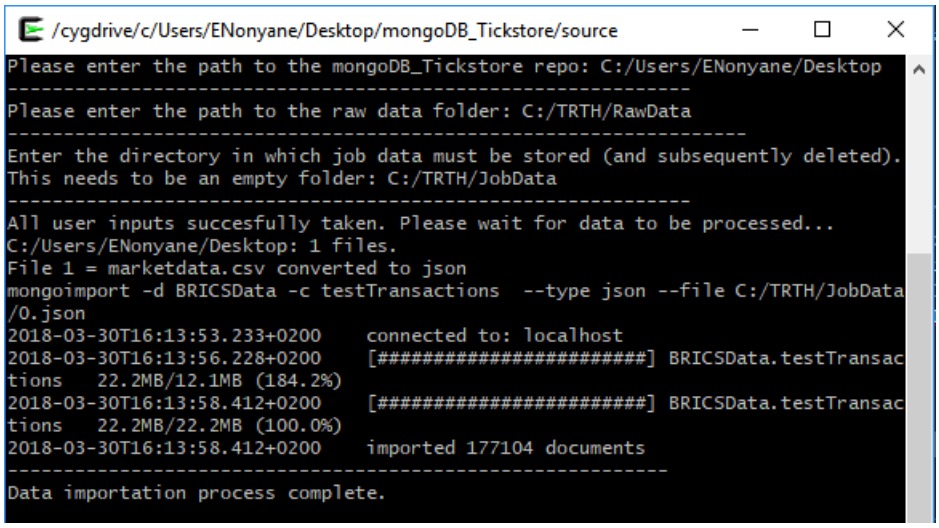
Although using C++ may seem inefficiency (after all, mongoimport can import file system data straight in a database with very little hassle) importing the data into the database this way gives us the opportunity to wrangle the data into a format that's easier for MongoDB to digest. The benefit of this is that it increases our data query ability (i.e we can query microsecond as opposed to daily data) and it also increases the speed at which we can query the database.

Step 5: import the data into the database

The next step is to import the market data into the database. This entails navigating the source folder in the repository and executing the command:

```
python3 importmongo.py
```

Upon executing this command, the user will get a prompt to specify the local directory of the *mongoDB_Tickstore* repository. Our repository was located on C:\Users\ENonyane\Desktop, thus, our input to this prompt was C:/Users/ENonyane/Desktop (note that forward slashes in the input). Next, the user will get a second prompt requesting for the directory of the raw data folder. Since we moved our market data to the directory C:\TRTH\RawData, our input to the second prompt was C:/TRTH/RawData (again, note the forward slashes). Lastly, the user will get a prompt requesting an empty directory to where the job data will be stored. Our input to this was C:/TRTH/JobData. If the data import process is successful, the user should get outputs similar to the ones shown in the below screen-shot.



Step 6: building indexes

After importing the data into the database, the next step is to build indices for the values of certain fields in order to expedite the rate at which we can query data. We build indices for the values of the field `DateTime` in the database. This was done by executing the command:

```
db.testTransactions.ensureIndex({"DateTime": 1})
```

in the mongo shell. This concludes the database development process. Methods described in section B.5 be used to query data from the database and save the queried data to a MATLAB workspace for processing.

Step 5: data analysis and visualization

With the database complete, the market data can be visualised and analysed. This requires the use of an application programming interface (API) similar to the one described in section B.5; this is provided in the *mongoDB_Tickstore* repository. As an example, the MATLAB script `visualiseOrderBook.m` (provided in the *Data analysis* folder of the repository) visualises the transactions and quote prices of the stock in the market data sample for the period between 12:30 till 13:30 on July 16, 2015. The produces visualizations are illustrated in figure B.2 and B.3.

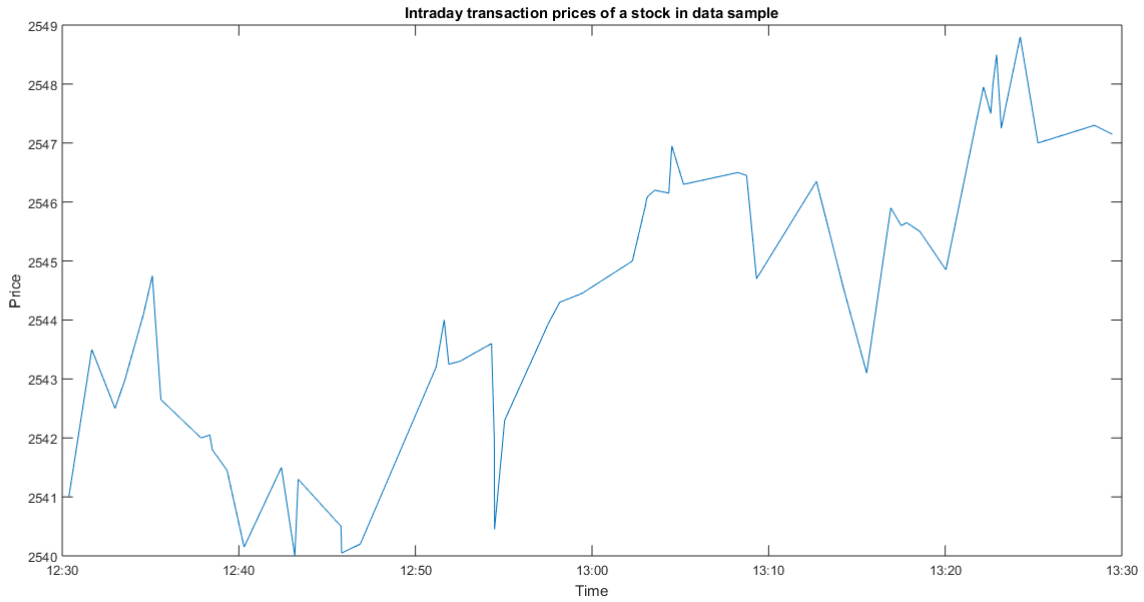


Figure B.2: Transaction prices of the stock in the market data sample provide in the repository *mongoDB_Tickstore* (availabe at https://github.com/Telmakaza/mongoDB_Tickstore). Here, the prices are for the period between 12:30 and 13:30 on July 16, 2015.

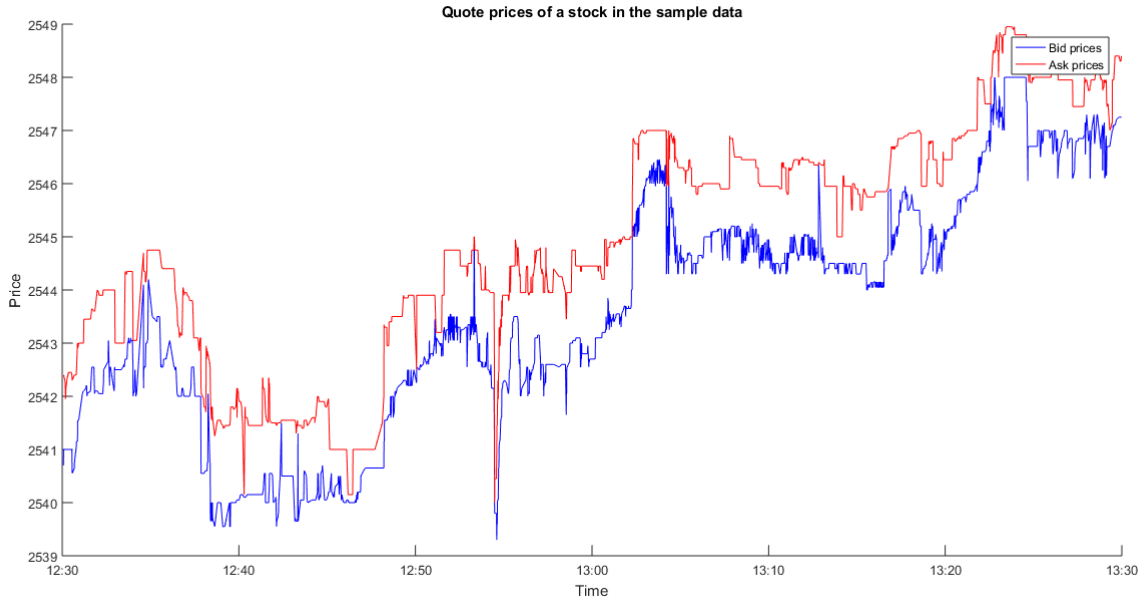


Figure B.3: Transaction prices of the stock in the market data sample provide in the repository *mongoDB_Tickstore* (availabe at https://github.com/Telmakaza/mongoDB_Tickstore). Here, the prices are for the period between 12:30 and 13:30 on July 16, 2015.

Appendix C

Source code

C.1 Database implementation 1: data import script

```
%% Import data into a MongoDB database.
% This script imports the contents of csv files into a collection called
% "JSETransactions" inside a MongoDB database called "BRICSData".

% Clear workspace
clc
clear

% Change current directory to data files directory
datafilepath = 'C:\ZMarket Data';
cd(datafilepath)

% Extract data file names
datafiles = dir(datafilepath);
datafiles = datafiles(~ismember({datafiles.name},{'.','..'}));
filenames = {datafiles.name};

% Get number of files
N = numel(filenames);

% Set collection and database name
Collection = 'JSETransaction';
Database = 'BRICSData';

% Set execution string parameters and options
exStrParameters = ['mongoimport -d',  Database,  ' -c ',...
  Collection, ' --type csv --file '];

exStrOptions = [' --ignoreBlanks  --fields "#RIC,Date[G],Time[G],GMT Offset',...
    'Type,Price,Volume,Market VWAP,Bid Price,Bid Size,Ask Price,Ask"']

% Loop over files
parfor j = 1:N

    % Construct data importation execution string
    exStr = [exStrParameters, filenames{j}, exStrOptions];

    % Execute execution string
    dos(exStr)
end
```

C.2 Database implementation 1: data extraction API

```
function  Data = extractfromMongoDBcsvPR(RIC, Collection, Date,...
 tempCSVoutput, fieldFile)
%EXTRACTFROMMONGODB  extract data from a MongoDB database called
%  BRICSData to MATLAB workspace.

% RIC - Stock ticker.
% Collection - Name of MongoDB collection where data is located.
% Date - Extraction date.
% tempCSVoutput - full file name of temporary csv output.
% fieldFile - Specifies extraction fields.

% Construct the execution string for mongoexport
[executionString] = ['"mongoexport.exe" ' ...,
    '--host 146.141.**.*** --port 27017 ', ...
    '--authenticationDatabase BRICSData ',...
    '-u **** ',...
    '-p **** ',...
    '-d BRICSData ', ...
    '-c ', Collection, ' ' ...
    '-q ', ...
    '"{''#RIC'': ''', RIC, ''', ''Date[G]'': ''' Date ''' }" ', ...
    '--type=csv ', ...
    '--fieldFile ', ...
    '"', fieldFile, '" ', ...
    '--out ', ...
    '"', tempCSVoutput, '"'];

% Execute mongoexport, data stored temporarily on disk in CSV file
dos(executionString,'-echo');

% Read CSV file into MATLAB workspace, remove extraneous quotations in
% string columns
tempData = textscan(fopen(tempCSVoutput),...
    '%q %q %q %q %q %q %q %q %q %q','delimiter', ',' ,'headerlines',1);
temp = str2double(cat(2,tempData{5:end}));
temp(isnan(temp)) = 0;
Data = [cat(2,tempData{1:4}),num2cell(temp)];

% Delete temporary CSV file
fclose('all');
delete(tempCSVoutput);
end
```

Note that for access control reasons, the user credentials have been censored with asterisks in the above script.

C.3 Database implementation 2: data import program

C.3.1 importdata.py

```
# importdata.py -- import TRTH data into BRICSData database
import os
from multiprocessing import Pool

exchanges = ["B3", "EGX", "JSE", "MOEX", "NSE", "SSE"]

def importmarketdata(market):
    os.system("C:\\Users\\ENonyane\\Documents\\DataEngineering\\bin\\main " + market)

if __name__ == '__main__':
    with Pool() as p:
        p.map(importmarketdata, exchanges)
```

C.3.2 main.cpp

```
/* DataEngineering/main.cpp -- Convert TRTH csv files into json files and import
to mongoDB database*/

#include "dsfunprot.h"
#include <iostream>
using std::cout;
using std::endl;

int main(int argc, char ** market)
{

    // Get filenames
    datafilenames rawData = getfilenames(market[1]);

    // Import convert files to json and import them to a mongoDB database;

    // Get number of files
    int n  = rawData.numfiles;
    cout << market[1] << ": " << n << " files." << endl;
    for(int j = 0; j < rawData.numfiles; j++)
    {
        toJSON(rawData.filenames[j], market[1], j);
        cout << "File 1 = " << rawData.filenames[j] << " converted to json" << endl;
    }

    // Call python import script
    string exstr =
        string("python3 C:/Users/ENonyane/Documents/")
            + string("DataEngineering/source/callmongoimport.py")
            + string(market[1]);
    system(exstr.c_str());

}
```

C.3.3 csvtojson.cpp

```
// csv_tojson.cpp -- Defines function that converts TRTH csv file to a JSON format
#include <fstream>
#include <string>
#include <stdio.h>
#include "dsfunprot.h"
#include <iostream>

using std::ofstream;
using std::ifstream;
using std::string;

// Bring rawdatapath and jobdatapath variables into scope
//extern string rawdatapath;
//extern string jobdatapath;

string rawdatapath = "C:/TRTH/RawData/";
string jobdatapath = "C:/TRTH/JobData/";
static char ch;

// define dummy variable to take inputs
string dummy;

inline void getwrite(const char * event, std::istream & inObj, std::ostream & outObj,
char delim)
{// Get input from input file and write to outputfile

    getline(inObj, dummy, ',');
    outObj << event << ":" << "\"" << dummy << "\"" << delim;
}

inline void getwritenum(const char * event, std::istream & inObj,
 std::ostream & outObj, char delim)
{// Get input from input file and write to outputfile

    getline(inObj, dummy, ',');
    dummy.empty() ? dummy = "0": dummy;
    outObj << event << ":" << dummy << delim;
}

string toJSON(string filename, const char * market, int filenum)
{
    // Declare file input and output objects
    ifstream fin;
    ofstream fout;

    // Associate file output object with an output file
    string outfile = jobdatapath + market + string("/") + std::to_string(filenum)
    + string(".json");
    fout.open(outfile);

    // Associate file input object with an input file
    string infile = rawdatapath + market + string("/") + filename;
    fin.open(infile);

    // Remove header from input stream
```

```
getline(fin, dummy);

// Read input from input file and write it to output json file
if(string(market) != "B3")
{
    while(fin.good())
    {
        fout << "{";
        getwrite("RIC", fin, fout, ','); // Get and write RIC

        // Declare variables for date, time gmtoffset and date-time
        string Date, Time, gmtOffset, DateTime;

        // Get date time and write to file
        getline(fin, Date, ',');
        getline(fin, Time, ',');
        getline(fin, gmtOffset, ',');
        DateTime = toISO(Date, Time, gmtOffset);
        fout << "DateTime:" << "{$date:" << "\""<< DateTime << "\"" << "}" << ",";

        // Get transaction an quote prices and volumes and write to json file
        getwrite("Type", fin, fout, ',');
        getwritenum("Price", fin, fout, ',');
        getwritenum("Volume", fin, fout, ',');
        getline(fin, dummy, ','); // Remove market vwap from input stream
        getwritenum("BidPrice", fin, fout, ',');
        getwritenum("BidVolume", fin, fout, ',');
        getwritenum("AskPrice", fin, fout, ',');
        getwritenum("AskVolume", fin, fout, '}');
        fout << '\n';

        // Remove excess data from input stream
        getline(fin, dummy);
    }
}
else
{
    while(fin.good())
    {
        fout << "{";
        getwrite("RIC", fin, fout, ','); // Get and write RIC

        // Declare variables for date, time gmtoffset and date-time
        string Date, Time, gmtOffset, DateTime;

        // Get date time and write to file
        getline(fin, Date, ',');
        getline(fin, Time, ',');
        getline(fin, gmtOffset, ',');
        DateTime = toISO(Date, Time, gmtOffset);
        fout << "DateTime:" << "{$date:" << "\""<< DateTime << "\"" << "}" << ",";

        // Get transaction an quote prices and volumes and write to json file
        getwrite("Type", fin, fout, ',');
        getline(fin, dummy, ','); // Remove "EX/Cntrb.ID" from input stream
        getwritenum("Price", fin, fout, ',');
        getwritenum("Volume", fin, fout, ',');
        getline(fin, dummy, ','); // Remove "Market VWAP" from input stream
```

```
            getline(fin, dummy, ','); // Remove "Buyer ID" from input stream
            getwritenum("BidPrice", fin, fout, ',');
            getwritenum("BidVolume", fin, fout, ',');
            getline(fin, dummy, ','); // Remove "No. Buyer" from input stream
            getline(fin, dummy, ','); // Remove "Seller ID" from input stream
            getwritenum("AskPrice", fin, fout, ',');
            getwritenum("AskVolume", fin, fout, '}');
            fout << '\n';

            // Remove excess data from input stream
            getline(fin, dummy);
        }
    }

    fin.close();
    remove(infile.c_str());

    fout.close();

    return outfile;
}
```

C.3.4 dsfunprot.h

```
//dsfunprot.h -- define "datafiles" struct and prototype functions

#include <iostream>
#include <string>
#include <fstream>
#include <string>
using std::string;

#ifndef DATAFILENAME_STRUCT_
#define DATAFILENAME_STRUCT_

struct datafilenames
{
    int numfiles;
    string filenames[100];
};

// Prototype functions
datafilenames getfilenames(const char *);
string & toISO(string &, string &, string &);
string toJSON(string, const char *, int);

#endif
```

C.3.5 toisodate.cpp

```
// toisodate.cpp -- convert date and time stamps into an isodate-like format
#include <iostream>
#include <string>

using std::string;

inline string & toDate(string & date, const char* month)
{// Convert "DD-MMM-YYYY" to "YYYY-MM-DD"
    date = date.substr(7, 4) + "-" + string(month) + "-" + date.substr(0, 2);
}

string & toISO(string & date, string & time, string & gmtOffset)
{
    if(gmtOffset.length() > 0)
    {
        switch (date[5])
        {
            case 'B': toDate(date, "02"); break; // Convert "FEB" to "02"

            // Convert "MAR" (resp. "APR") to "03" (resp. "04")
            case 'R': date[3] == 'M' ? toDate(date, "03") : toDate(date, "04"); break;


            case 'Y': toDate(date, "05"); break; // Convert "MAY" to "05"

            // Convert "JAN" (resp. "JUN") to "01" (resp "06")
            case 'N': date[4] == 'A'? toDate(date, "01"): toDate(date, "06"); break;

            case 'L': toDate(date, "07"); break; // Convert "JUL" to "07"

            case 'G': toDate(date, "08"); break; // Convert "AUG" to "08"

            case 'P': toDate(date, "09"); break; // Convert "SEP" to "09"

            case 'T': toDate(date, "10"); break; // Convert "OCT" to "10"

            case 'V': toDate(date, "11"); break; // Convert "NOV" to "11"

            case 'C': toDate(date, "12"); break; // Convetr "DEC" to "12"

        }

        // Switch sign of GMTOffset
        gmtOffset[0] == '+' ?  gmtOffset[0] = '-': gmtOffset[0] = '+';

        // Change date to ISODate
        date += "T" + time + gmtOffset.substr(0, 1) + "0" + gmtOffset.substr(1,1)
         + "00";
        return date;
    }

    return date;

}
```

C.3.6 getfilenames.cpp

```
// getfilenames.cpp -- reads file names from a csv file and saves them to a struct

#include <stdio.h>
#include <string.h>
#include <fstream>
#include "dsfunprot.h"
using std::ifstream;
using std::ofstream;
using std::string;


datafilenames getfilenames(const char * Market)
{
    // Allocate memory space
    char exstr[1024];

    // Copy execution string to allocated memory space
    strcpy(exstr,
    "python3 C:/Users/ENonyane/Documents/DataEngineering/source/getfilenames.py ");
    strcat(exstr, Market);
    system(exstr);

    // Read file names from input file
    ifstream infnames;
    string file_on_sys = string("filenames") + Market + string(".txt");
    infnames.open(file_on_sys);

    // Count number of files and store names
    string files[100];
    int counter = 0;
    while(infnames.good())
    {
        getline(infnames, files[counter]);
        ++counter;
    }

    // Declare datafilenames struct
    datafilenames dfstruct;

    // Initailise datafilenames struct
    dfstruct.numfiles = counter - 1;
    for(int i = 0; i < counter; i++)
    {
        dfstruct.filenames[i] = files[i];
    }

    infnames.close();
    remove(file_on_sys.c_str());

    return dfstruct;
}
```

C.3.7 getfilenames.py

```
# getfilenames.py -- get file names of raw data and import them to c++
import os
import sys

# Get file names
Market = sys.argv[1];
datafilepath = 'C:/TRTH/RawData/' + Market
filenames = os.listdir(datafilepath)

# Write file names to text files
fobj = open('filenames' + Market + '.txt', 'w')
for name in filenames:
    fobj.write(name + "\n")

fobj.close()
```

C.3.8 callpythonimport.py

```
# callmongoimport.py -- call mongoimport in parallel
import os
import sys
from multiprocessing import Pool

# Take user input
Market = sys.argv[1]

# Set data path and get files in directory
datafilepath = 'C:/TRTH/JobData/' + Market
filenames = os.listdir(datafilepath)

# Import files in parallel
def mongoimportcmd(filename):
    exstr =
    "mongoimport -d BRICSData -c " + Market + "Transactions " + "--type json --file "
    + datafilepath + "/" +filename
    os.system(exstr)
    fullfilename = datafilepath + "/" + filename
    os.remove(fullfilename)

if __name__ == '__main__':
    with Pool() as p:
        p.map(mongoimportcmd, filenames)
```

C.4 Database implementation 2: data extraction API

```
function data = extractfromMongoDBcsvPR2(ticker, collection, startDateTime,...
 endDateTime, tempcsvout, fieldFile)

% Convert Start/End dates to ISO date number (milliseconds since Unix
% epoch)
[unixEpoch] = datenum(1970,1,1,0,0,0);
[unixStartDateTime] = round((datenum(startDateTime) - unixEpoch) * 86400000);
[unixEndDateTime] = round((datenum(endDateTime) - unixEpoch) * 86400000);

% Construct the execution string for mongoexport
[executionString] = ['"mongoexport.exe" ',...
    '-d BRICSData ', ...
    '-c ', collection, ' ' ...
    '-q ', ...
    '"{''RIC'': ''', ticker,...
     ''', ''DateTime'': {''$gt'': new Date(', num2str(unixStartDateTime),...
      ') , ''$lte'': new Date(', num2str(unixEndDateTime),')}}" ', ...
    '--type=csv ', ...
    '--fieldFile ', ...
    '"', fieldFile, '" ', ...
    '--out ', ...
    '"', tempcsvout, '"'];
% Execute mongoexport, data stored temporarily on disk in CSV file
dos(executionString);

data = textscan(fopen(tempcsvout),...
 '%s %s %q %f %f %f %f %f %f', 'delimiter', ',', 'headerlines', 1);

% Change dates to datenum
data = cat(2, data{1:3}, num2cell([data{4:end}]));
data(:,3) = num2cell(datenum(data(:,3), 'yyyy-mm-ddTHH:MM:SS.FFF'));
fclose('all');

delete(tempcsvout);
end
```

C.5 Data cleaning

```
function cleanData = CleanCompt(Data, Market)
%CLEANCOMPT — clean and compact raw data from BRICSData database

% Data — Raw data set
% Market — Market to which raw data belongs (e.g 'JSE', 'BSE')


% Remove Auctions
Data(strcmp(Data(:,4),'Auction'),:) = [];
Data(strcmp('',Data(:,1)),:) = [];

% Get continuous trading start time + 10 minutes and end time — 10 minutes,
% in GMT time, for market.
switch Market
    case 'JSE'
        STime = '07:10:00.000';
        ETime = '14:50:00.000';

    case 'BSE'
        STime = '03:55:00.000';
        ETime = '09:50:00.000';

    case 'BVSP'
        STime = '12:10:00.000';
        ETime = '18:45:00.000';

    case 'ESE'
        STime = '08:10:00.000';
        ETime = '12:20:00.000';

    case 'MOEX'
        STime = '07:10:00.000';
        ETime = '15:29:00.000';

    case 'NSE'
        STime = '06:40:00.000';
        ETime = '11:50:00.000';

    case 'SSE'
        STime = '01:40:00.000';
        ETime = '03:20:00.000';

        STime2 = '05:10:00.000';
        ETime2 = '06:50:00.000';
end


% Clean out events that are not part of continous trading
if strcmp(Market,'SSE')

    % Convert the start and end times into date numbers
    STime = datenum(STime);
    ETime = datenum(ETime);
```

```
STime2 = datenum(STime2);
ETime2 = datenum(ETime2);


% Convert all event times into date numbers
Data(:,3) = num2cell(datenum(Data(:,3)));

%-----------------------------------------------------------------

%-----------------Session 1 data cleaning-----------------------


% Clean trade sesssion 1 data
tempDataSet1 = Data;

% Extract event times
eventTimes = [tempDataSet1{:,3}]';

% Clear data from 10 minutes just after opening (in ses 1)
tempDataSet1(eventTimes<STime,:) = [];

% Update event times
eventTimes =[tempDataSet1{:,3}]';

%  Clear data 10 minutes before closing in ses 1
tempDataSet1(eventTimes>ETime,:) = [];

% Create an index for all session 1 data
tradeSessionIndex1 = ones(size(tempDataSet1,1),1);

%-----------------------------------------------------------------

%-----------------Session 2 data cleaning-----------------------

% Clear trade session 2 data
tempDataSet2 = Data;

% Update event times
eventTimes = [tempDataSet2{:,3}]';

% Clear data 10 from 10 minutes after opening in ses 2
tempDataSet2(eventTimes<STime2,:) = [];

% update event times
eventTimes = [tempDataSet2{:,3}]';

% Clear data from 10 minutes before clossing in ses 2
tempDataSet2(eventTimes>ETime2,:) = [];

% Create an index for session 2 data
tradeSessionIndex2 = 2*ones(size(tempDataSet2,1),1);


% Reconstrcut the data cell
Data = [tempDataSet1;tempDataSet2];

% Make an index to track trades of the two different trade
% periods
```

```
    tradeSessionIndex = [tradeSessionIndex1;tradeSessionIndex2];
    Data = [Data, num2cell(tradeSessionIndex)];


else

    % Convert start and end dates into date numbers
    STime = datenum(STime);
    ETime = datenum(ETime);

    % Convert all the event times into date numbers
    Data(:,3) = num2cell(datenum(Data(:,3)));

    % Extract the trade times
    eventTimes = [Data{:,3}]';

    % Clear events 10 minutes after opening
    Data(eventTimes<STime,:) = [];

    % Update event times
    eventTimes = [Data{:,3}]';

    % Clear events 10 minutes before closing
    Data(eventTimes>ETime,:) = [];

    % Create an index for all of the session one events
    tradeSessionIndex = num2cell(ones(size(Data,1),1));

    % Reconstruct the data cell
    Data = [Data, tradeSessionIndex];
end


% Separate quotes from trades
dataQ = Data(strcmp(Data(:,4),'Quote'),:);
dataT = Data(strcmp(Data(:,4),'Trade'),:);


% "Compact" trades and quotes
if ~isempty(dataT) && ~isempty(dataQ)

    % Populate missing data with appropriate data
    for i = 2:size(dataQ,1)

        for Column = 7:10
            if dataQ{i,Column} == 0
                dataQ{i,Column} = dataQ{i-1,Column};
            end
        end
    end

    % Extract the quote times
    quoteTimes = [dataQ{:,3}]';

    % Get list of quotes with unique timestamp
    [~,uniquerowindex] = unique(quoteTimes,'last');

    % Clear quotes with non-unique time stamps
```

```
    dataQ = dataQ(uniquerowindex,:);

    % Clean nonsensical quotes and trades
    dataQ(cell2mat(dataQ(:,7))<=0,:) = [];
    dataQ(cell2mat(dataQ(:,8))<=0,:) = [];
    dataT(cell2mat(dataT(:,6))<=0,:) = [];

    if ~isempty(dataT) && ~isempty(dataQ)

        % Get trade timestamps
        tradesTimes = [dataT{:,3}]';

        % VWAP compact trades with the same timestamp
        col5Prices = cell2mat(dataT(:,5));
        col6Vols = cell2mat(dataT(:,6));

        for j = 1:numel(tradesTimes)
            P = col5Prices(tradesTimes == tradesTimes(j));
            Q = col6Vols(tradesTimes == tradesTimes(j));
            dataT{j,5} = P'*Q/sum(Q);
            dataT{j,6} = sum(Q);
        end

        % Remove trades with the same timestamp
        [~,utindex] = unique(tradesTimes,'first');
        dataT = dataT(utindex,:);

        % Return cleaned data
        cleanData = sortrows([dataT;dataQ],3);

    end
end
```

C.6 Plot intraday prices/volumes

```
function plotOrderBook(RIC, Market, data, userChoice)
%PLOTORDERBOOK - Visulise the top of the book.
%   PLOTORDEBOOK(RIC, Market, data, userChoice) plots the level 1 bid, ask
%   and trade prices or volumes, for a  single trading day, against time.
%   If userChoice is 'Prices', plot of prices is generated. Else, if
%   userChoice is 'Volume', plot of volume is generated

% RIC - Stock ticker.
% Market - Marketer to which stock with ticker RIC belongs.
% data - cleaned data set generated by using CleanCompt()  on output cell
% array of extractfromMongoDBcsvPR()

% Extract trade and quote prices and volumes.
bidPrices = cell2mat(data(:,7));
askPrices = cell2mat(data(:,9));
bidVolumes = cell2mat(data(:,8));
askVolumes = cell2mat(data(:,10));
tradePrices = cell2mat(data(:,5));
tradeVolumes  = cell2mat(data(:,6));

% Set GMT Offset
switch Market
    case 'BSE'
        GMTOffset = '05:30';
    case 'BVSP'
        GMTOffset = '02:00';
    case 'ESE'
        GMTOffset = '02:00';
    case 'JSE'
        GMTOffset = '02:00';
    case 'MOEX'
        GMTOffset = '03:00';
    case 'NSE'
        GMTOffset = '03:00';
    case 'SSE'
        GMTOffset = '08:00';
end

if strcmp(Market,'BVSP')
    GMTOffset = -datenum(GMTOffset);
else
    GMTOffset = datenum(GMTOffset);
end

% Add GMT offset event times.
times = cell2mat(data(:,3)) + GMTOffset;

% Change zeros into Nans
bidPrices(bidPrices==0) = nan;
askPrices(askPrices==0) = nan;
tradePrices(tradePrices==0) = nan;
bidVolumes(bidVolumes==0) = nan;
askVolumes(askVolumes==0) = nan;
tradeVolumes(tradeVolumes==0) = nan;
```

```
% Create new figure
figure
hold on

switch userChoice

    case 'Prices'

        % Plot the order book prices
        bprice = plot(times, bidPrices,'.', 'color', 'b', 'markersize', 15);
        aprice = plot(times, askPrices,'.', 'color', 'r', 'markersize', 15);
        tprice = plot(times, tradePrices,'.', 'color', 'w', 'markersize', 5);
        set(gca, 'color', 'k')
        datetick(gca,'x','HH:MM')
        xlabel('Time')
        ylabel('Price (in local currency)')
        title(['Quote and trade prices of ', RIC, ' on ', Date])
        plegend = ...
        legend([aprice, bprice, tprice], 'Ask price', 'Bid price', 'Trade price');
        set(plegend, 'color', [.5 .5 .5])


    case 'Volumes'

        % Plot order book volumes
        bvol = plot(times, bidVolumes, '.b', 'markersize', 5);
        avol = plot(times, askVolumes, '.r', 'markersize', 5);
        tvol = plot(times, tradeVolumes, '.w', 'markersize', 5);
        set(gca, 'color', 'k')
        set(gca,'yscale','log')
        datetick(gca, 'x', 'HH:MM')
        xlabel('Time')
        ylabel('Volume')
        title(['Quote and trade volumes of ', RIC, ' on ', Date])
        vlegend =...
         legend([avol, bvol, tvol], 'Ask Volume', 'Bid Volume', 'Trade Volume');
        set(vlegend, 'color', [.5 .5 .5])
end
```

C.7 Plot market transactions

```
function  p = plotTrades(data, Market)
%PLOTTRADES - plot the prices of trades with respect to time directly from
%the orderbook of a particular stock
% PLOTTRADES(Stock) produces a two dimensional plot of trade prices
% against (local) time.

% RIC - Stock ticker.
% Market - Marketer to which stock with ticker RIC belongs.
% data - cleaned data set generated by using CleanCompt()  on output cell
% array of extractfromMongoDBcsvPR()

% Extract trade prices, volumes and times
timePriceVol = cell2mat(data(strcmp(data(:,4), 'Trade'), [3 5 6]));

% Set GMT Offset
switch Market
    case 'BSE'
        GMTOffset = '05:30';
    case 'BVSP'
        GMTOffset = '02:00';
    case 'ESE'
        GMTOffset = '02:00';
    case 'JSE'
        GMTOffset = '02:00';
    case 'MOEX'
        GMTOffset = '03:00';
    case 'NSE'
        GMTOffset = '03:00';
    case 'SSE'
        GMTOffset = '08:00';
end

if strcmp(Market,'BVSP')
    GMTOffset = -datenum(GMTOffset);
else
    GMTOffset = datenum(GMTOffset);
end

% Offset date to local time
timePriceVol(:,1) = timePriceVol(:,1) + GMTOffset;

% Bin trading volumes
N = 4; % Number of bins
binEdges = zeros(N,1);
for j = 1:N
    binEdges(j) = quantile(timePriceVol(:,3), j/N);
end
binidx = binData(timePriceVol(:,3), binEdges);

% Generate plot
figure(1)
hold on
C = rand(1,3);
for i = 1:length(timePriceVol)
```

```
    if numel(binidx) < i
        break
    end

    if binidx(i) == 1
        M = 1;
    elseif binidx(i) == 2
        M  = 5;
    elseif binidx(i) == 3
        M = 10;
    elseif binidx(i) == 4
        M = 20;
        if timePriceVol(i,3) > 10*std(timePriceVol(:,3))+ mean(timePriceVol(:,3))
            M = 50;
        end
    end
    p = plot(timePriceVol(i,1), timePriceVol(i,2), '.', 'Color', C, 'MarkerSize', M);
end
datetick(gca,'x','HH:MM')
xlabel('Local time')
ylabel('Price (in local currency)')


function binidx = binData(dataSet, binEdges)
binidx = zeros(numel(dataSet),1);
for i = 1:numel(dataSet)
    for j = 1:numel(binEdges)
        if j == 1
            if dataSet(i) <= binEdges(j)
                binidx(i) = j;
            end
        elseif j~=1
            if dataSet(i) > binEdges(j-1) && dataSet(i) <= binEdges(j)
                binidx(i) = j;
            end
        end
    end
end
```

C.8 Data extraction, cleaning and visualization test script

```
%% Test script for data extraction, cleaning and visualization
% Author: E. Nonyane
% Date: November 16, 2017

%% Clear workspace
clc
clear
close all

%% Set initial parameters
Market  = 'BVSP';
Collection = [Market, 'Transactions'];
Date = '15-APR-2015';
RIC = 'CPLE6.SA';
tempCSV = 'C:\Users\MSS_Master\JobData\temp.csv';
fieldFile = 'C:\Users\MSS_Master\Documents\MATLAB\ENonyane_PR\MongoExp\ff2.csv';

%% Exctarct data
data = extractfromMongoDBcsvPR2(RIC, Collection, Date,...
 tempCSV, fieldFile);

%% Clean data
cleanData = CleanCompt(data, Market);

%% Plot
plotOrderBook(RIC, Market, cleanData, Date, 'Prices')
plotTrades(cleanData, Market)
```

C.9 Single stock price response class skeleton

```
classdef PriceResponse < handle
    % PriceResponse -  Class defenition for price response objects.
    % This class is used to calculate the price response of stocks from
    % interday transaction data. It attempts to encapsulate the data and
    % methods used to calculate the bare impact and price impact of stocks.

     %% Class attributes
    properties
        RIC % Stock ticker
        Marker % Stock exchange at which stock with ticker RIC is traded belongs.
        CL % Autocorrelation of trade signs.
        RL % Price response coefficient.
        G0 % Bare response function.
        DL % Mean squared displacement of the price.
        CLV % C_1(\ell).
        BPIM % Buyer Price Impact Matrix. Contains average price shifts due to buyer
        % initiated trades and corresponding normalized volumes.
        SPIM % Seller Price Impact Matrix. Contains average price shifts due to seller
        % initiated trades and corresponding normalized volumes.

    %% Class procedures
    methods
        %% PriceRespnse constructor
        function obj = PriceResponse(Tickers,Spacing,Mkt)

        %% Data cleaning method
        function CleanCompt(varargin)

        %% Trade classification algorithm
        function Classf(obj)

        %% Method for calculating daily parameters
        function obj = DailyParameters(obj,Collection,SDate,EDate,...
                tempCSVoutput, transactionsFieldFile, userChoice)

        %% Method for calculating the daily average price response
        function TakeAverages(obj)

        %% Method for calculating the price impact function
        function CalPrIM(obj)

        %% Method for calibrating parameters
        function obj = calibPR(obj)

        %% Method for calculating the alpha and lambda constants of price impact
        function  calBetaLambda(obj)

        %% Method to calibrate bare response
        function  calibrateG0(obj)

        %% Method to calibrate C(L)
        function  calibrateCL(obj)

    end
```

C.10 Grouped price response class skeleton

```
classdef  GroupedPriceResponse < handle
    % GrupedPriceResponse - Class definition of GroupedPriceResponse Object
    % This class is used to calculate the price response of grouped stocks. It
    % attempts to encapsulate the data and methods used to calculate the
    % the price impact and bare impact functions of grouped stocks.
    % Each instance of this class represents a trading day.

    properties
      Date % The date of a specific day .
      Tickers % The tickers of a group of stocks.
      GdPB % Buyer price chanage.
      GdPS % Seller price change.
      GBV % Volumes corresponind to the price change vector GdPB.
      GSV % Volumes corresponding to the price change vector GdPS.
      GRL % Price response coefficient of grouped stocks.
      GDL % Square root of normalized mean squared displacement of the price.
      GCL % Trade sign autocorrelation function of grouped stocks.
      GG0 % Estimate of bare response function of grouped stocks
      Market % Stock exchange at which stocks with RICs "Tickers" are traded.
      GTB % Trade times of buyer price impacts.
      GTS % Trade times of seller price impacts.
      GBPIM % Grouped buyer price impact matrix.
      GSPIM % Grouped seller price impact matrix.
    end

     %% GroupedPriceResponse procedures
    methods

        %% GroupedPriceResponse constructor method
        function obj = GroupedPriceResponse(varargin)

        %% Method for calculating stock parameters
        function obj = stockParameters(obj,Collection,Date,tempCSVoutput,transff)

    end
```

C.11 Calculating price response in parallel

```
%% Clear Workspace
clc
clear
close all

%% Congifure paths
addpath('C:\Users\MSS_Master\Documents\MATLAB\ENonyane_PR\Classes');
addpath('C:\Users\MSS_Master\Documents\MATLAB\ENonyane_PR\functions');

%% Initialize parameters.
MARKETS = {'BVSP', 'BSE', 'ESE', 'JSE', 'MOEX', 'NSE', 'SSE'};

startDate = {'01-Jan-2010','01-Jan-2011','01-Jan-2012','01-Jan-2013',...
        '27-Jan-2014','01-Jan-2015','01-Jan-2016'};

endDate = {'31-Dec-2010','31-Dec-2011','31-Dec-2012','31-Dec-2013',...
        '31-Dec-2014','31-Dec-2015','10-May-2016'};

FieldFile = 'C:\Users\MSS_Master\Documents\MATLAB\ENonyane_PR\MongoExp\ff.csv';

%% Loop over markets
for m = 1:7

    Market = MARKETS{m};

    % Import report csv
    marketCell = importMarketCSV(Market);

    % Extract stock RIC from report csv
    Tickers = unique(marketCell(:,1));

    % Eliminate "#RIC" and index from stock list
    Tickers = Tickers(3:end);

    % Remove items which are not stock from ticker list
    if any(strcmp(Market,'MOEX') | strcmp(Market,'BVSP'))
        Tickers = Tickers(2:end);
    end

    % Get number of tickers
    n = numel(Tickers);

    % Instantiate price response object array
    Stocks = PriceResponse(Tickers,'logSpace',Market);

    % Set collection name
    Collection = [Market,'Transactions'];

    % Populate the 'Analysis' PriceResponse property
    for i = 1:numel(Tickers)
          Stocks(i).Analysis = 'Single';
    end

    % Loop over years
```

```
    for y = 1

        % Set start and end dates
        SDate = startDate{y};
        EDate = endDate{y};

        % Calculate price response in parallel
        parfor i=1:numel(Tickers)
            tempOut = ['C:\JobData\enonyane\tempdata\temp',Market,num2str(i),'.csv'];
            Stocks(i) = Stocks(i).DailyParameters(Collection,SDate,EDate,...
                tempOut,FieldFile,'Full Price Response');
        end

        % Save output
        save(['C:\Users\MSS_Master\Documents\MATLAB\ENonyane_PR\Market Information\',
        Market,'\SingleStock\',Market,'1Y',SDate(end-3:end),'.mat'],'Stocks')

        % Clear stock variable
        clear Stocks

    end
end
```

C.12 Stock binning algorithm

```
function binidx = binByNumTrades(obj)
% BINBYNUMTRADES group stocks by total number of trades per day.
%   binidx = BINBYADTV(obj) outputs a (n x 1) matrix, binidx. Each element
%   of binidx correspons to a "number of trades" bin. Here, obj is an object
%   array of type PriceResponse.

% Get number of objects
n = numel(obj);

% Initialize bin index vector
binidx = zeros(n,1);

% Extract number of trades per day for each stock
Tradesperday = zeros(n,1);

for k = 1:n
    Tradesperday(k) = sum(obj(k).tradesPerDay);
end

tradesPerBin = sum(Tradesperday)/4;

% Create matrix that tracks Stocks
    stockMatrix = [(1:n)', Tradesperday];

    % Sort stockMatrix by ADTV
    stockMatrix = sortrows(stockMatrix,2);
    sortedADTV = stockMatrix(:,2);

    % Initialise Sum variable and loop indices
    S = sortedADTV(1);
    i = 1;
    j = 1;

    % Initialise index vector and Sum vector
    sumInBin = zeros(3,1);
    binEdges = zeros(3,1);


    % Get bin edges
    while i < n

        S = S + sortedADTV(i+1);

        if S >= tradesPerBin
            sumInBin(j) = S;
            binEdges(j) = i;

            % update loop index
            j = j+1;

            if i<=n-2
                S = sortedADTV(i+2);
                i = i+1;
            end
```

```
    end

    i = i + 1;
end

% Get size of binEdges vector
binEdges(binEdges==0) = [];
m = numel(binEdges);

% Populate binidx vector
for j = 1:m
    if j ==1
        binidx(1:binEdges(j)+1) = j;
    else
        binidx(binEdges(j-1)+2:binEdges(j)+1) = j;
    end
end

% unsort stocks

sortedBinMatrix = [binidx,stockMatrix(:,1)];
originalOrderedMatrix = sortrows(sortedBinMatrix,2);

binidx = originalOrderedMatrix(:,1);
binidx(binidx==0) = 3;
```

Appendix D

Machine Specifications

D.1 Database server: Bella

- Processor: Intel(R) Core(TM) i7 CPU x980 @ 3.33GHz
- RAM: 24,0GB
- System Type: 64-bit Operating System, x64-based processor

D.2 Computation solution: Superella

- Processor: Intel(R) Xeon CPU E5-2683 v4 @ 2.10 GHz (2 processors)
- RAM: 256GB
- System Type: 64-bit Operating System, x64-based processor

D.3 Computation solution ii: DESKTOP-3SEBVL2

- Processor: Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz 3.60 GHz
- RAM: 16.0GB
- System Type: 64-bit Operating System, x64-based processor

Bibliography

[1] P. Jain and F. Johnson. Trading Technology and Stock Market Liquidity: A Global Perspective. 2006. Available at: https://papers.ssrn.com (last accessed on: November 01, 2017).

[2] J. Hasbrouck. *Empirical Market Microstructure*. Oxford University Press, Inc., 2007.

[3] S. Hruska. Competing models for market data dissemination: a comparison of stock and futures markets. 2002. Available at: https://www.mercatus.org (last accessed on: November 01, 2017).

[4] WysInfo Docwebs. Doves and Pigeons in History. Available at: http://www.wysinfo.com (last accessed on: November 01, 2017).

[5] FXCM. Evolution of the market place: from open outcry to electronic trading. Available at: https://www.fxcm.com (last accessed on: November 01, 2017).

[6] M. Phister. Quotron II: An early multiprogrammed multiprocessor for the communication of stock market data. 1989. Available at: http://ieeexplore.ieee.org (last accessed on: November 02, 2017).

[7] R. Sinn. Reminiscences of a stock quotation system: a real story of Ultronic Corporation. 2009. Available at: https://www.inductive.com (last accessed on: November 02, 2017).

[8] C. Goodhart and M. O'Hara. High frequency data in financial markets: Issues and applications. *Journal of Empirical Finance*, 4:73–144, 1997.

[9] F. Heatheway. Competition, incentives and innovations in the great American marketplace. In R. Schwartz, J. Bryne, and E. Stempel, editors, *Rapidly Changing Securities Markets*, chapter 3, pages 33–34. Springer, 2017. Available at: https://books.google.co.za (last accessed on: November 02, 2017).

[10] T. Robards. Private network is established for trading institutional stocks. In F. Alt, editor, *Advances in Computers*, page 111. Academic Press, New York, 1970.

[11] Thompson Reuters. Thompson Reuters Tick History, 2012. Available at: https://financial.thomsonreuters.com (last accessed on: November 04, 2017).

[12] NYSE. NYSE Market Data, 2016. Available at:http://www.nyxdata.com (last accessed on: November 04, 2017).

[13] University of the Witwatersrand. Reduced JSE fees have a cost, 2017. Available at: https://www.wits.ac.za (last accessed on: November 04, 2017).

[14] OECD. Data-Driven Innovation, Big Data for Growth and Well-Being, 2015. Available at: http://www.oecd-ilibrary.org (last accessed on: November 04, 2017).

[15] E. Zivot. Analysis of high frequency financial data: models methods and software. Part I: descriptive analysis of high frequency financial data with S-PLUS. 2005. Available at: https://faculty.washington.edu (last accessed on: November 04, 2017).

[16] R. Almgren, C. Thum, E. Hauptmann, and H. Li. Direct estimation of market impact. 2005. Available at: http://www.courant.nyu.edu (last accessed on: November 04, 2017).

[17] R. Engle. The econometrics of ultra-high frequency data. 1996. Available at: ttp://www.nber.org (last accessed on: November 05, 2017).

[18] F. Lillo, J. Farmer, and R. Mantegna. Master curve for price-impact function. *Nature*, 421:129–130, 2003. Available at: http://tuvalu.santafe.edu (last accessed on: November 05, 2017).

[19] Who is hosting this. S-PLUS programming resources. Available at: https://www.whoishostingthis.com (last accessed on: November 11, 2017).

[20] I. Giampaoli, W. Lon Ng, and N. Constantinou. Analysis of ultra-high-frequency financial data using advanced Fourier transforms. *Finance Research Letters*, 2009. Available at: http://www.sciencedirect.com (last accessed on: November 05, 2017).

[21] J. Glattfelder, A. Dupuis, and R. Olsen. Patterns in high-frequency FX data: discovery of 12 empirical scaling laws. *Quantitative Finance*, 11(4):559–614, 2010. Available at: http://www.tandfonline.com (last accessed on: November 05, 2017).

[22] B. Du Preez, D. Wilcox, and T. Gebbie. JSE market microstructure. 2015. Available at: http://wiredspace.wits.ac.za (last accessed on: November 05, 2017).

[23] M. Harvey, D. Hendricks, D. Wilcox, and T. Gebbie. Deviations in expected price impact for small transaction volumes under fee restructuring. *Physica A*, 471, 2017. Available at: http://www.sciencedirect.com (last accessed on: November 04, 2017).

[24] Google. Limit order definition. Available at: https://www.google.co.za.

[25] J. Bouchaud. Price impact. 2009. Available at: https://arxiv.org (last accessed on: November 06, 2017).

[26] A.Kraus and H.Stoll. Price Impact of Block Trading on the New York Stock Exchange. *The Journal of Finance*, 27(3):569–588, 1972. Availabe at: http://www.jstor.org (last accesed on: October 23, 2017).

[27] A. Kyle. Continuous auction and insider trading. *Econometrica*, 53(6):1315–1335, 1985. Available at: https://www.jstor.org (last accessed on: November 07, 2017).

[28] M. Potters and J. Bouchaud. More statistical properties of order books and price impact. *Physica A*, 324:133–140, 2003. Available at: https://www.sciencedirect.com (last accessed on: November 07, 2017).

[29] J. Bouchaud, Y. Gefen, M. Potters, and M. Wyart. Fluctuations and response in financail markets: the subtle nature of 'random' price changes. *Quantitative Finance*, 4:176–190, 2003. Available at: http://www.tandfonline.com (last accessed on: August 19, 2017).

[30] J. Hausman and A.Lo. An ordered probit analysis of transaction stock prices. *Journal of Financial Economics*, 31:319–379, 1992. Available at: http://www.sciencedirect.com (last accessed on: October 23, 2017).

[31] L. Chan and J. Lakonishok. The behavior of stock prices around institutional trades. *The Journal of Finance*, 50(4):1147–1174, 1995. Available at: http://www.jstor.org (last accessed on: November 07, 2017).

[32] F. Lillo, D. Farmer, and R. Mantegna. Single curve collapse of the price impact function for the New York Stock Exchange. 2002. Available at: https://arxiv.org.

[33] C. Lee and M. Ready. Inferring trade direction from intraday data. *The Journal of Finance*, 46(2):733–746, 1991. Available at: http://www.jstor.org (last accessed on: November 08, 2017).

[34] G. Iori, N. Daniels, J. Farmer, L. Gillemot, S. Krishnamurthy, and E. Smith. Analysis of price impact function in order-driven markets. *Physica A*, 324(1-2):146–151, 2003. Available at: http://www.sciencedirect.com.

[35] M. Lim and R. Coggins. The immediate price impact of trades in the Australian Stock Exchange. *Quantitative Finance*, 5(4):365–377, 2005. Available at: http://dx.doi.org/10.1080/14697680500151400 (last accessed on: August 19, 2017).

[36] F. Patzelts and J. Bouchaud. Universal scaling and nonlinearity of aggregate price impact in financial markets. 2017. Available at: https://arxiv.org/abs/1706.04163 (last accessed o: March 10, 2019).

[37] A. Kyle and A. Obizhaeva. The Market Impact Puzzle. 2018. Available at: https://papers.ssrn.com (last accessed o: March 10, 2019).

[38] R. Almgren and Chriss. Optimal execution of portfolio transactions. 2000. Available at: https://www.courant.nyu.edu (last accessed on: November 09, 2017).

[39] G. Huberman and W. Stanzl. Price manipulation and quasi-arbitrage. *Econometrica*, 72(4):1247–1275, 2004. Available at: https://www.jstor.org (last accessed on: November 09, 2017).

[40] F. Lillo and J. Farmer. The long memory of efficient market. *Studies in Nonlinear Dynamics & Econometrics*, 8(3), 2004. Available at: http://tuvalu.santafe.edu (last accessed on: November 21, 2017).

[41] H. Laurence. "Liquidity" and "Transaction Costs". *Southern Economic Journal*, 32(1):43–48, 1965. Available at: https://www.jstor.org (last accessed o: January 12, 2017).

[42] T. Gebbie. Email conversation. 2018.

[43] D. Suciu. Semi-Structured Data Model. In *Encyclopedia of Databases*, pages 2601–2605. Springer US, 2009.

[44] D. Hendricks and E. Nonyane. QuERILab MongoDB Technical Documentation. 2017.

[45] Johannesburg Stock Exchange. JSE Trading and Information Overview, 2016. Available at: https://www.jse.co.za.

[46] Acervo Digital. Bovespa - digital memo. Available at: https://translate.googleusercontent.com (last accessed: August 17, 2017).

[47] High Beam Research. Rio de Janeiro stock exchange sold off to BM&F, 2002. Available at: https://www.highbeam.com (last accessed: August 17, 2017).

[48] S. Nyasha and N. Ohiambo. The Brazilian Stock Market Development: A Critical Analysis and the Prospects during the past 50 years . *Risk Governance & Control: Financial Markets & Institutions*, 2013. Available at: http://www.virtusinterpress.org (last accessed: August 17, 2017).

[49] BM&F Bovespa. BM&F Bovespa Puma Trading System, 2016. Available at: http://www.bmfbovespa.com.br (last accessed on: November 13, 2017).

[50] BM&F Bovespa. BM&FBOVEPSA Operational Procedure Manual - Bovespa Segment, 2014. Available at: http://www.bmfbovespa.com.br (last accessed on: November 13, 2017).

[51] BM&F Bovespa. Market maker, 2016. Available at: http://www.bmfbovespa.com.br (last accessed on: November 17, 2017).

[52] BM&F Bovespa. Market maker, 2016. Available at: http://www.bmfbovespa.com.br (last accessed on: November 13, 2017).

[53] BM&FBOVESPA. Bovespa Index (ibovespa), 2016. Available at: http://www.bmfbovespa.com (last accessed on: November 11, 2017).

[54] MB&FBOVESPA. Bovespa Index methodology, 2014. Available at: http://www.bmfbovespa.com.

[55] J. Bouchaud, D. Farmer, and F. Lillo. How markets slowly digest changes in supply and demand. 2008. Available at: https://arxiv.org/pdf/0809.0822.pdf (last accessed on: September 25, 2017).

[56] BSE Ltd. BSE-Heritage. Available at: http://www.bseindia.com (last accessed on: August 14, 2017).

[57] Mumbai Commercial Capital of India. Bombay stock exchange. Available at: http://www.mumbai.org.uk (last accessed on: August 14, 2017).

[58] BSE Ltd. Bse-Milestones. Available at: http://www.bseindia.com (last accessed on: August 15, 2017).

[59] BSE Ltd. Bse-Introduction. Available at: http://www.bseindia.com (last accessed on: August 15, 2017).

[60] The Economic Times. Bse becomes world's fastest stock exchange: Ashishkumar chauhan, 2015. Available at: http://economictimes.indiatimes.com (last accessed on: August 15, 2017).

[61] BSE Ltd. Session timings. Available at: http://www.bseindia.com (last accessed:August 22, 2017).

[62] BSE Ltd. Equity trading. Available at: http://www.bseindia.com (last accessed on: August 23, 2017).

[63] The Egyptian Exchange. History, 2017. Available at: http://www.egx.com.eg (last accessed on: August 18, 2017).

[64] The Egyptian Exchange. Faqs, 2017. Available at: http://www.egx.com.eg (last accessed on: August 18, 2017).

[65] The Egyptian Exchange. Trading system, 2017. Available at: http://www.egx.com (last accessed on: August 05, 2017).

[66] Investor Relations Nasdaq. NASDAQ OMX and the Egyptian Exchange extend technology agreement, 2012. Available at: http://ir.nasdaq.com (last accessed on: August 05, 2017).

[67] The Egyptian Exchange. Egyptian Exchange trading hours, 2017. Available at: http://www.egx.com.eg (last accessed on: November 13, 2017).

[68] The Egyptian Exchange. Egyptian Exchange trading mechanisms and rules, 2017. Available at: http://www.egx.com.eg (last accessed on: November 13, 2017).

[69] The World Federation of Exchanges. Monthly Reports. Available at: https://www.world-exchanges.org (last accessed on: August 04, 2017).

[70] M. Lukasiewicz. From Diamonds to Gold: The Making of the Johannesburg Stock Exchange. *Journal of Southern African Studies*, 2017. Available at: http://www.tandfonline.com (last access on: July 29, 2017).

[71] South African History Online. Colonial history and development of johannesburg, 2011. Available at: http://www.sahistory.org.za (last accessed: August 13, 2017).

[72] M. Lukasiewicz. Reinvestigating Southern African stock markets: The making of the Johannesburg Stock Exchange. 1880-1889. 2013.

[73] P. Vecchiato. JSE prepares for change to new trading systems, 2002. Available at: http://www.itweb.co.za (last access on: July 29, 2017).

[74] Brand South Africa. SA debut for LSEs TradElect, 2007. Available at: https://www.brandsouthafrica.com (last access on: July 29, 2017).

[75] Johannesburg Stock Exchange. JSE's new Equity Trading Platform in SA after Decade in London, 2012. Available at: http://ir.jse.co.za (last access on: July 29, 2017).

[76] JSE Ltd. The JSE Top 40 Index, 2013. Available at: https://www.jse.co.za (last accessed on: November 13, 2017).

[77] Bloomburg Markets. JSE Top 40 Quote, 2017. Available at: https://www.bloomberg.com (last accessed on: November 13, 2017).

[78] Moscow Exchange. History, 2014. Available at: https://www.moex.com (last accessed on: August 19, 2017).

[79] G. Gustafson. *Capitalism Russian-Style*. Cambridge University Press, 2003.

[80] Moscow Exchange. Opening auction — Moscow Exchange, 2017. Available at: http://www.moex.com (last accessed on: November 13, 2017).

[81] Moscow Exchange. Main market (T+2) — Moscow Exchange, 2017. Available at: http://www.moex.com (last accessed on: November 13, 2017).

[82] Moscow Exchange. Closing auction — Moscow Exchange, 2017. Available at: http://www.moex.com (last accessed on: November 13, 2017).

[83] Moscow Exchange. Indices — Moscow Exchange, 2017. Available at: http://www.moex.com (last accessed on: November 13, 2017).

[84] Nairobi Securities Exchange. History of the NSE, 2017. Available at: https://www.nse.co.ke (last accessed on: November 12, 2017).

[85] Nairobi Securities Exchange. Nairobi Securities Exchange Equity Securities Trading Rules, 2017. Available at: https://www.nse.co.ke (last accessed on: November 13, 2017).

[86] Nairobi Stock Exchange. Ground rules nse 20 share index constituent companies selection criteria, 2017. Available at: ttps://www.nse.co.ke (last accessed on: November 13, 2017).

[87] J. Chavis. The history of China's Stock Market, 2017. Available at: https://bizfluent.com (last accessed on: November 12, 2017).

[88] W. Fan. Shanghai stock market 1871-1940, 2010. Available at: https://www.slideshare.net (last accessed on: November 12, 2017).

[89] Shanghai Stock Exchange. Historical events, Shanghai Stock Exchange, 2015. Available at: http://english.sse.com.cn (last accessed on: November 12, 2017).

[90] Shanghai Stock Exchange. Trading Rules of Shanghai Stock Exchange, 2006. Available at: http://english.sse.com.cn (last accessed on: November 17, 2017).

[91] Shanghai Stock Exchange. SSE 50 Methodology. Available at: http://www.sse.com.cn (last accessed on: November 14, 2017).

[92] D. Hendricks, D. Wilcox, and T. Gebbie. An online adaptive learning algorithm for optimal trade execution in high-frequency markets. Wits University. 2016.

[93] Cygwin. Cygwin, 2018. Available at: `https://www.cygwin.com/` last accessed on: March 29, 2018.