High frequency dynamics of order flow

Xiaowei Zheng

Wadham College
University of Oxford

MSc in Mathematical and Computational Finance Thesis
Trinity 2013

Acknowledgements

I would like to thank my supervisor, Dr. Samuel N. Cohen, for his invaluable support and guidance. In addition, I would like to thank my coursemate, Kakin Chan for his helpful suggestions during my writing. Finally, special thanks are due to my parents - Jinsen Zheng and Feiming Pan.

Contents

List of Figures

.

Abstract

In this paper, we focus on the high frequency dynamics of limit order flow and market order flow. We compared the fitting performance of different models for the inter-arrival time of the order flow, including exponential distribution, gamma distribution and power law. We then studied the dependence of the placement of these two order flows, which can be captured by the self-excitation effect and mutual-excitation effect of Hawkes process. We also introduced a new model which combines the Hawkes features with the gamma distribution.

Key words: High frequency dynamics; order flow; market microstructure; maximum likelihood estimation; Hawkes process; Hawkes-Gamma distribution.

Chapter 1

Introduction

1.1 Limit order book

The *continuous double auction* is the most widely used trading mechanism in today's fast-paced financial world. It features an electronic limit order book, and enables financial markets to operate continuously. Limit order book serves as a medium to match buyers and sellers of a specified asset, and hence the asset price can be determined by the execution of trades. The auction is called "continuous double" since traders can submit orders in form of *bids* (*i.e.* buy orders) as well as *asks* (*i.e.* sell orders) at any point in time. A bid order (*resp.* ask order) specifies the price and the quantity of the asset to be purchased (*resp.* to be sold). Outstanding orders are then stored in limit order book in bid and ask queues, which are sorted by the price of bid order and the price of ask order respectively. The highest bid is on top of the bid queue while the lowest ask is on top of the ask queue.

Traders are allowed to place two types of orders: *limit order* and *market order*. Limit order is the type that does not cross the top of the book on the opposite side, and thereby will not trigger an immediate transaction. (*Farmer et al.*, 2005 [8]) Intuitively, a limit bid order is placed with a price less than the best ask price while a limit ask order is placed with a price greater than the best bid price. Market order is another type of order that does cross the top of the book on the opposite side, and thus triggers an immediate transaction. In other words, a market bid order causes an immediate execution of a limit ask order while a market ask order causes an execution of a limit bid order. In this context, limit orders accumulate in both bid and ask queues, while market orders remove limit orders from the queues due to transactions.

Apart from the placement of orders, the cancellation of limit orders is another commonplace event. It generally implies that the owner of a limit order is no longer willing to offer a trade at the stated price, however it is also employed as part of strategy in high frequency trading. Due to the explosion of high frequency trading during recent years, cancellations take place far more frequently than before. (*Harris et al.*, 2002 [11])

From a theoretical point of view, patient traders place limit orders while impatient traders place market orders. The limit order book is then used to transfer the liquidity provided by patient traders to impatient traders. In reality, most traders use a combination of limit orders and market orders; they select their actions for each situation based on their individual needs at that time. (*Anand et al.*, 2005 [1]) Nevertheless, technical traders or arbitrageurs generally prefer to place market orders (due to their short-term trading strategy) and fundamental traders or portfolio managers prefer to place limit orders (due to their long-term trading strategy). The interactions between the heterogeneous agents make the dynamics of limit order book remarkably complicated.

Several critical characteristics of limit order book are defined as follows. At time t, the highest price of limit buy orders is called the *best bid* (denoted by $b(t)$ in the sequel) while the lowest price of limit sell orders is called the best ask (denoted by $a(t)$). In the queue of limit buy orders, from the highest price to the lowest price, limit buy orders are called to be placed in different *price levels*; that is, level 1 represents the highest bid, level 2 represents the second highest bid, and so forth. Similar terminology is adopted in the queue of limit sell orders. Thereby, best bid and best ask are the corresponding price in level 1. The price slippage of limit order book is defined as the *bid-ask spread* (denoted by $s(t)$), which measures the gap between best bid and best ask, *i.e.* $s(t) = a(t) - b(t)$. The mid-quote price $m(t)$ is defined as the average of best bid and best ask, *i.e.* $m(t) = (a(t) + b(t))/2$. In addition, there are also two precision parameters in limit order book. One is the minimum order size, which is the smallest quantity that can be traded in the market. Another one is the tick size, which is the smallest permissible price interval between different orders. In modelling, the measure of relative price is widely used. For a limit order with price p, the relative price is the difference of order price and the top of the book; that is, the relative price of limit buy order is $\delta^b(p) = b(t) - p$ and the relative price of limit ask order is $\delta^a(p) = p - a(t)$. An illustration of the characteristic of limit order is shown in Figure 1.1.

1.2 Literature review

Limit order book contains a host of valuable information in terms of the current market status. It is a result of local interactions between many heterogeneous agents in the market. In recent years, there are many research on the features of limit order book. In general, they can be classified into two categories.

The first category is to study the statistical regularities of limit order book. *Gould et al.* (2011 [9]) summarises a detailed list of empirical observations in limit order book. He

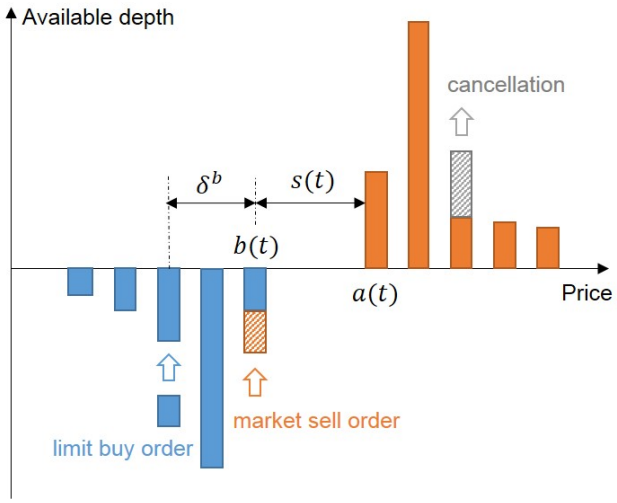

Figure 1.1: Illustration of limit order book

suggested that different empirical studies often present conflicting conclusions. This may be a result of many reasons, including different characteristics in different markets and the evolution of trader's strategies over time. However, there are still some stylized facts that were found common in different markets, such as the power-law fits of the distribution of order size and the distribution of relative price, the hump shape of mean relative depth profiles, the long memory in order flow, and so forth (see more details in [9]).

The second category is to model the dynamics of limit order book using ideas from economics, physics, mathematics and statistics. A clear understanding of limit order book can help us optimise order placement strategies and minimise market impact. *Kirilenko et al.* (2011 [13]) also showes that limit order book offers an insight into how to assess market stability. At present, there are two main schools in limit order book modelling. One is agent-based approach, which assumes the perfect rationality of market participants who aim to maximise their own utility. Another one is zero-intelligence modelling, which focuses on the aggregate effect of individual agents' actions without considering their own motivations.

Since limit order books offer a very complex scenario, there are many limitations when using an agent-based approach. *Parlour et al.* (2008 [18]) outlines the general procedures of the agent-based approach, which assumes that rational investors select portfolio strategies to maximise personal utility, subject to budget constraints, by using limit orders or market orders. The main limitation is that in reality, investors often intend to employ trading strategies consisting of both limit orders and market orders instead of submitting orders for exact quantities at exact prices (*Gould et al.*, 2011 [9]). Hence, zero-intelligence modelling is closer to the real dynamics of limit order book. It assumes that order arrivals and can-

cellations follow some stochastic processes. The parameters can thereby be calibrated from historical data. Generally, it is first trained with a subset of available data in-sample, and then evaluated with the rest of data out-of-sample (*cross-validation*).

Smith et al. (2002 [19]) suggests the arrival of limit orders, the cancellations, and the arrival of market orders follow three homogeneous Poisson process, with three different constant parameters. He then proposes a dimensional analysis to forecast the mean spread and the diffusion rate using these three parameters. *Cont et al.* (2008 [7]) follows a similar assumption of Smith's model, that is, the arrivals and cancellations of orders follow homogeneous Poisson process. However, he incorporates the stylized facts that order arrival rates depend on relative price (*Bouchaud et al.*, 2002 [3]; *Zovko et al.*, 2002 [22]). He suggests that the arrivals and cancellations of orders with different relative price follow a homogeneous Poisson process with different parameters, where the parameters satisfy a power law fit. He then adopts the birth-death process to compute some conditional probabilities, such as probability of increase in mid-price and probability of order execution before mid-price moves.

Biais et al. (1995 [2]) proposes the strong clustering effect of market events: arrival of market buy order, arrival of limit buy order within the spread, and cancellation of active sell order. *Bouchaud et al.* (2009 [4]) suggests that the event clustering effect was primarily in that traders often strategically split large orders into small chunks so as to minimise market impact. *Bowsher* (2003 [5]) proposes to use stochastic intensity other than constant intensity to model the clustering effect. *Toke et al.* (2012 [21]) focuses on modelling trades-through, which are transactions that reach at least the second level of limit orders in an limit order book.

Due to the failure of simple zero-intelligence models to capture the dependencies between various types of orders, such as clustering effect, the interplay between liquidity taking and providing, in this study we focus on the interdependence of limit order flow and market order flow. We attempt to capture these interrelationships within a more developed zero-intelligence model.

Section 2 outlines the mathematical framework of order flow modelling. In Section 3, we empirically study some stylised facts of order flow and fit the inter-arrival time of order flows by several standard fittings, including exponential distribution, gamma distribution and power law. In Section 4, we introduce Hawkes process to capture the dependence of limit order flow and market order flow. We also introduce Hawkes process with gamma distribution. Finally Section 5 concludes.

Chapter 2

Order flow modelling

2.1 Mathematical framework

Intuitively, the dynamics of the limit order book are driven by the order flow. The mathematical framework of order flows under zero-intelligence modelling is employed in this study. In this framework, two main types of agents are involved: impatient agents and patient agents. Impatient agents place market orders following a stochastic process while patient agents place limit orders following another stochastic process. Unexecuted limit orders are assumed to be cancelled also following a stochastic process. All orders are assumed to be of unit size. This setting contributes to the formation of the main market events:

- the arrival of a limit buy order at price $p < a(t)$ increases the depth of bid queue at price p by one unit;
- the arrival of a limit sell order at price $p > b(t)$ increases the depth of ask queue at price p by one unit;
- the arrival of a market buy order decreases the depth of ask queue at price $a(t)$;
- the arrival of a market sell order decreases the depth of bid queue at price $b(t)$.
- the cancellation of an outstanding limit buy order at price $p < b(t)$ decreases the depth of bid queue at price p by one unit;
- the cancellation of an outstanding limit sell order at price $p > a(t)$ decreases the depth of ask queue at price p by one unit;

Note that the incoming order flow can be represented as a counting process. The homogeneous Poisson process is then a common assumption in zero-intelligence models ([7, 19]). Empirically, a symmetry of ask and bid order placement is observed, which is also the case in our data set (see Section 3.2). Thereby, we reduce these six types of market events to be three. Mathematically, we assume that

- the counting process of limit order placement at a relative price of i[1] ticks follows a Poisson process with intensity $\lambda^L(i)$;
- the counting process of market order placement follows a Poisson process with intensity λ^M;
- the counting process of limit order cancellation at a relative price of i ($i \geq 0$) ticks follows a Poisson process with intensity $\lambda^C(i)\,x$, where x is the depth of outstanding orders;

The intensity of limit order cancellation is assumed to be proportional to the outstanding limit orders, which is fit for empirical results, as well as our data set (see Section 3.2). According to *Bouchaud et al.* (2002 [3]), a power law fits the intensity of limit order placement depending on the relative price of i ticks: $\lambda^L(i) \propto i^{-\gamma}$ ($i \geq 1$).

2.2 Dependence of order flow

Bouchaud et al. (2009 [4]) demonstrates that the order flow is highly persistent long-memory process due to order splitting. Hence several phenomenon might be expected: the arrival of a limit order (resp. market order) expedites the next limit order (resp. market order), which is called the self-excitation effect; the more frequent arrival of limit orders may expedite the arrival of market orders, and vice versa, which is called the mutual-excitation effect. In this section, we introduce the Hawkes process to capture the dependence of order flow.

2.2.1 Hawkes process

Hawkes process is a generalized version of Poisson process with stochastic intensity, which was first introduced by Hawkes (1971 [12]). We start by defining a simple M-variate point process. Suppose T_i and Z_i are two random variables on probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where T_i is the time of occurence of the i-th event and $Z_i = \{1, 2, ..., M\}$ is the type of the i-th event. Note that $T_i \leq T_{i+1}$ and $\lim_{i\to\infty} T_i = +\infty$. Hence, the counting process of type m ($m \in \{1, 2, ..., M\}$) is defined as $N_t^m = \sum_{T_i \leq t} \mathbb{I}_{\{Z_i = m\}}$, and then $N = \left(N^1, N^2, ..., N^M\right)$ is a M-variate simple point process.

Let $\mathcal{F}_t = \sigma(N_t)$ be the natural filtration, then the intensity process λ_t^m associated with counting process N_t^m satisfies

$$\mathbb{E}\left[N_t^m - N_s^m | \mathcal{F}_s\right] = \mathbb{E}\left[\int_s^t \lambda_u^m du | \mathcal{F}_s\right] \tag{2.1}$$

[1] Note that i can be negative here, which implies the placement of limit order within bid-ask spread.

where $0 \leq s \leq t$. Notice that λ^m refers to the conditional probability that an event of type m occurs per unit of time. λ^m can then be modelled as a stochastic intensity process

$$\lambda_t^m = \lambda_0^m + \sum_{n=1}^{M} \int_0^t G_{t-s}^{mn} dN_s^n \tag{2.2}$$

where G_t^{mn} is a kernel function. In this study, we assume the stochastic intensity is mean-reverting. Therefore, an exponentially-decaying kernel $G_t^{mn} = \alpha^{mn} e^{-\beta^{mn} t}$ is employed. Note that parameters α^{mn} and β^{mn} imply the influence (scale and decay speed respectively) of the past events of type n on the m-th intensity process. In particular, $m = n$ indicates the self-excitation effect and $m \neq n$ indicates the mutual-excitation effect.

We employ a Hawkes process to capture the features of two order flows (limit order flow and market order flow) in the limit order book. The counting process of limit order placement is denoted by N^L associated with stochastic intensity process

$$\lambda_t^L = \lambda_0^L + \int_0^t \alpha_{LL} e^{-\beta_{LL}(t-s)} dN_s^L + \int_0^t \alpha_{ML} e^{-\beta_{ML}(t-s)} dN_s^M \tag{2.3}$$

where parameters α_{LL} and β_{LL} capture the self-excitation effect of limit order flow while α_{ML} and β_{ML} capture the mutual-excitation effect of market order flow on limit order flow. Likewise, the counting process of market order placement is denoted by N^M associated with stochastic intensity process

$$\lambda_t^M = \lambda_0^M + \int_0^t \alpha_{MM} e^{-\beta_{MM}(t-s)} dN_s^M + \int_0^t \alpha_{LM} e^{-\beta_{LM}(t-s)} dN_s^L \tag{2.4}$$

where parameters α_{MM} and β_{MM} capture the self-excitation effect of limit order flow while α_{LM} and β_{LM} capture the mutual-excitation effect of limit order flow on market order flow.

2.2.2 Stationarity condition

For an M-variate Hawkes process $N = \left(N^1, N^2, ..., N^M\right)$, it is important to assume stability over the course of time. The stationarity of a Hawkes process is then defined as the joint distribution of $(N_{t_1+t}^{i_1} - N_{s_1+t}^{i_1}, N_{t_2+t}^{i_2} - N_{s_2+t}^{i_2}, ..., N_{t_r+t}^{i_r} - N_{s_r+t}^{i_r})$ (where $s_j \leq t_j$; $i_j \in \{1, 2, ..., M\}$) is invariant under translation, namely the joint distribution does not rely on $t \in \mathbb{R}$.

Equation (2.2) is equivalent to the vector form $\lambda_t = \lambda_0 + \int_0^t G_{t-s} dN_s$ where $G_t = [G^{mn}]_{m,n=1,2,...,M}$. *Bremaud et al.* (1996 [6]) proposes a stationarity condition for Hawkes

process under vector form, which is the spectral radius of matrix

$$\Gamma = \int_0^\infty G_u du = \left(\frac{\alpha^{mn}}{\beta^{mn}}\right)_{m,n=1,2,\ldots,M} \tag{2.5}$$

is strictly less than 1.

For processes (2.3) and (2.4), the stationarity condition is simplified as

$$\frac{1}{2}\left(\frac{\alpha_{LL}}{\beta_{LL}} + \frac{\alpha_{MM}}{\beta_{MM}} + \sqrt{\left(\frac{\alpha_{LL}}{\beta_{LL}} - \frac{\alpha_{MM}}{\beta_{MM}}\right)^2 + 4\frac{\alpha_{LM}\alpha_{ML}}{\beta_{LM}\beta_{ML}}}\right) < 1 \tag{2.6}$$

2.3 Estimation and simulation

2.3.1 Maximum likelihood estimation

The parameters of Hawkes process is generally estimated by maximum likelihood method. For an M-variate Hawkes process (Equation (2.2)), the log-likelihood $\log \mathcal{L}(N_t)$ can be written as the sum of the partial log-likelihood of type m, namely

$$\log \mathcal{L}(N_t) = \sum_{m=1}^{M} \log \mathcal{L}^m(N_t^m) \tag{2.7}$$

where $\log \mathcal{L}^m(N_t^m) = \int_0^t (1-\lambda_s^m)\, ds + \int_0^t \log \lambda_s^m dN_s^m$. *Ozaki* (1979 [17]) suggested a recursive method to simplify the calculation of maximum likelihood. Using his result, the partial log-likelihood of type m event can be calculated as

$$\begin{aligned} \log \mathcal{L}^m &= t(1-\lambda_0^m) - \sum_{t_i \le t} \sum_{n=1}^{M} \frac{\alpha^{mn}}{\beta^{mn}} \left(1 - e^{-\beta^{mn}(t-t_i)}\right) \\ &\quad + \sum_{t_i \le t} \log\left(\lambda_0^m + \sum_{n=1}^{M} \alpha^{mn} R_i^{mn}\right) \end{aligned} \tag{2.8}$$

where the recursive term R_i^{mn} satisfies $R_0^{mn} = 0$ and if $m \neq n$,

$$R_i^{mn} = e^{-\beta^{mn}(t_i - t_{i-1})} R_{i-1}^{mn} + \sum_{t_{i-1} \le t_k^n \le t_i} e^{-\beta^{mn}\left(t_i - t_k^n\right)} \tag{2.9}$$

if $m = n$,

$$R_i^{mn} = e^{-\beta^{mn}(t_i - t_{i-1})}\left(1 + R_{i-1}^{mn}\right) \tag{2.10}$$

The parameters can then be estimated by maximising Equation (2.8). On the grounds that it is not easy to obtain an explicit form of the solution to optimisation problem (2.8)

subject to the stationarity condition, a numerical maximisation algorithm is suggested to solve this problem (see Section 4.2).

After obtaining the maximum likelihood estimator (denoted by θ^{MLE}), it is also helpful to calculate the associated errors of MLE. A general theory suggests to use the inverse of Fisher information $I(\theta)$, namely

$$var\left(\theta^{MLE}\right) \approx \left(I(\theta)\right)^{-1} = \left(-\mathbb{E}\left[\frac{\partial^2 \log \mathcal{L}^m(\theta)}{\partial\theta_i \partial\theta_j} \mid_{\theta=\theta^{MLE}}\right]\right)^{-1} \tag{2.11}$$

For details, see Lehmann (1998 [14], page 463).

2.3.2 Simulation of Hawkes process

In order to test the fitting performance of Hawkes process, a simulation algorithm is required. *Ogata* (1981 [16]) proposed an algorithm to simulate an M-variate Hawkes process. The critical procedures of this algorithm are listed in Algorithm 1 (see more details in [16]). Suppose we intend to simulate a series of $\{t_i\}$ within $[0, T]$, which represents the time stamps of the event occurrence.

Algorithm 1 Simulation of multivariate Hawkes process

1. **Initialisation**:

   (a) **Initialisation of maximum intensity**: Let $i = i^1 = i^2 = ... = i^M = 1$ and $I^* = I^M = \sum_{m=1}^{M} \lambda_0^m$;

   (b) **First event**: Generate a uniform random variable $U \sim \mathcal{U}[0,1]$ and let $s = -\log U / I^*$. If $s > T$, then go to (a);

   (c) **Attribution test**: Generate $V \sim \mathcal{U}[0,1]$. Let $t_1^{n_0} = s$ where n_0 satisfies $I^{n_0-1} < VI^* \leq I^{n_0}$. Afterwards let $t_1 = t_1^{n_0}$.

2. **General routine**:

   (a) **Update of maximum intensity**: Let $i^{n_0} = i^{n_0} + 1$, $i = i + 1$ and $I^* = I^M + \sum_{m=1}^{M} \alpha^{mn_0}$;

   (b) **New event**: Generate a uniform random variable $U \sim \mathcal{U}[0,1]$ and let $s = s - \log U / I^*$. If $s > T$, then go to (a);

   (c) **Attribution test**: Generate $V \sim \mathcal{U}[0,1]$.

      i. If $VI^* \leq I^M$, then let $t_{i_{n_0}}^{n_0} = s$ where n_0 satisfies $I^{n_0-1} < VI^* \leq I^{n_0}$. Afterwards let $t_i = t_{i_{n_0}}^{n_0}$. Go to (a);

      ii. If $VI^* > I^M$, then let $I^* = I^M$ and go to (b).

Chapter 3

Stylised facts of order flow

3.1 Description of data set

In the sequel, our study is based on the order book of four futures market. The data is provided by *Man Investments*. It consists of order book data of four futures for the whole trading week from *27 October 2008* to *31 October 2008*. The instruments involved are as follows

- EDC: Eurodollar short term interest rate (STIR) future contract for December 2009, traded on the Chicago Mercantile Exchange (CME);
- EUL: Euribor STIR future contract for December 2009, traded on the London International Financial Futures and Options Exchange (LIFFE);
- SSL: Short-sterling STIR future contract for December 2009, traded on LIFFE.
- FTL: FTSE 100 stock index future contract for December 2008, traded on LIFFE;

For every order data set, at any point in time, it is comprised of 10 levels of bid price and the number of corresponding outstanding limit buy orders, 10 levels of ask price and the number of corresponding outstanding limit sell orders, as well as the last trading volume, the total trading volume and the open interest.

For Eurodollar STIR future contract (EDC), the data set merely contains one level of data, namely best bid, best ask and their corresponding depth. For Euribor STIR future contract (EUL), there are only two of five days (*27 October 2008* and *30 October 2008*) that contain continuous quotes for the major trading period. For short-sterling STIR future contract (SSL), the quotes were updated every ten seconds in the afternoon, which could not satisfy the requirement of this study, from a very short time horizon. Finally, we focused on the study of FTSE 100 stock index future contract (FTL), which contains continuous quotes of ten levels on both bid and ask side for a whole trading week. Since we are only interested in the period of brisk market activity, the order book data between 8.00 am and

7.00 pm is studied.

Before proceeding further on the study, we conducted an identification procedure in the first place for the order book data in order to reconstruct an event table which keeps track of intraday market events. Six types of market events are marked:

1. Submission of a new limit order;
2. Cancellation of a limit order (partial deletion of the outstanding limit orders at the same level);
3. Cancellation of a limit order (total deletion of the outstanding limit orders at the same level);
4. Execution against a visible limit order (partial depletion of the outstanding limit orders at the same level);
5. Execution against a visible limit order (total depletion of the outstanding limit orders at the same level);
6. Execution against a hidden limit order.

The events of type 6 correspond to the existence of *iceberg order*s[1] in the market. The fundamental approach to identify the submission and the cancellation of limit orders is to monitor the change of order quotes and depths at all the price levels on both bid and ask side. Notice that the increase of order depth can only be a result of the submission of a new limit order; nevertheless, the decrease of order depth can either be due to the cancellation of a limit order or the execution (*i.e.* the arrival of market order). The change of last trading volume is then employed to verity the real reason of the decrease. An event is marked as type 6, *i.e.* execution against a hidden limit order, when there is an increase in trading volume without a decrease in the order depth or a removal of a quote.

The basic descriptive statistics of the market events for FTSE 100 stock index future contract (FTL) during the active trading period (8.00 am to 7.00 pm) is shown in Table 3.1. It demonstrates that there are 56901 limit orders submitted every hour on average, which accounts for almost the half of all the market events in order book. The cancellations are also very frequent, with a similar intensity as limit order placements. However, the submission of market orders, or the executions against limit orders, is considerably less than the other two major sources of market events. Roughly, the submission of limit orders is over ten times frequent than the submission of market orders in our data set. In addition,

[1] This is a type of limit order that specifies a total size and price as well as a visible size. Only the visible size is shown in the limit order book.

over half of market orders are executed against a hidden limit order, which implies that iceberg orders are very common in this market.

Arrivals of limit order	# events of type 1 (hourly)	56901	56901
Cancellations	# events of type 2 (hourly)	40215	53252
	# events of type 3 (hourly)	13037	
Arrivals of market order	# events of type 4 (hourly)	1421	4792
	# events of type 5 (hourly)	869	
	# events of type 6 (hourly)	2502	

Table 3.1: Market events on order book data (FTL)

3.2 Relative price

Cont et al. (2008 [7]) suggests that limit order flow should be studied by different relative price, *i.e.* the distance to the top of the book on the same side in terms of ticks. Figure 3.1 shows the placement of limit buy orders and limit sell orders depends on the relative price. Evidently, the major activity of limit order placement occurs around the top of the book. There are also almost 15% of aggressive limit orders, which are limit orders placed within the bid-ask spread. *Bouchaud et al.* (2002 [3]) reports a power law fit of limit order placement. In our data set, the placement of limit orders at a relative price of 3 ticks and 4 ticks disobey the power law, which reveals that the power law is not always valid for all assets and all markets. In addition, a very strong symmetry of buy side and sell side is observed in our data set.

Apart from the limit order placement, the cancellations also depend on relative price. Figure 3.2 shows that the major cancellations of limit orders take place around the top of the book (over 50% within one tick of relative price). A similar strong symmetry is also shown for limit buy order cancellations and limit sell order cancellations. Besides, when comparing Figure 3.1 and Figure 3.2, there is a similar shape on the positive side, which justifies the assumption of Cont et al. that the cancellations are proportion of the outstanding orders at any point in time.

3.3 Exponential fitting

Cont et al. assumes that the counting process of limit order placement at a relative price of i ticks follows a homogeneous Poisson process with intensity $\lambda^L(i)$. Notice that the inter-arrival time of a Poisson counting process follows an exponential distribution. We studied the inter-arrival time of limit order placement at a relative price from 0 to 5 ticks. The empirical probability density functions are then shown in Figure 3.3. Unsurprisingly,

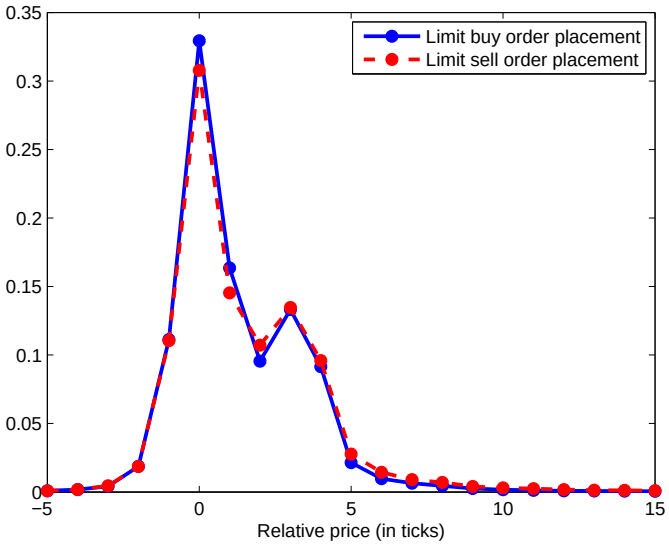

Figure 3.1: Limit order placement w.r.t. relative price

a very fast decay is observed in all the distributions. The log-log plot also shows a linear tail for all the distributions.

Following the classical assumption of zero-intelligence models (*Smith et al.*, 2002 [19]; *Cont et al.*, 2008 [7]), an exponential distribution is employed to fit the inter-arrival time of limit order placement. In our data set, the intensities $\lambda^L(i)$ at relative price of i ticks are estimated using maximum likelihood estimation. The results are shown in Table 3.2. We cannot recover the power law of intensities as *Bouchaud et al.* reported in 2002, which is mainly due to the outlier of intensity at a relative price of 3 ticks.

Relative price	0	1	2	3	4	5
Intensity λ^L	10.1318	8.3379	7.0547	7.3277	6.8042	6.4168

Table 3.2: Limit order placement intensity

In the meanwhile, the fitting distributions are demonstrated in Figure 3.4. It is obvious that the decay speed around zero of exponential distribution is not as fast as the empirical distribution. Moreover, the fitting distribution cannot reproduce the linear tails. These results suggest that the standard assumptions of zero-intelligence models are not well justified empirically in our data set.

As another standard assumption of zero-intelligence models, the arrival of market orders follows an exponential distribution as well. In our data, we also fitted an exponential

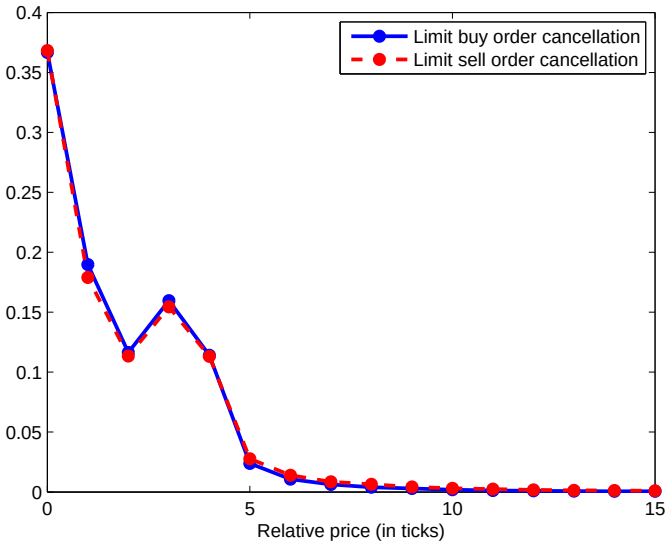

Figure 3.2: Limit order cancellaiton w.r.t. relative price

distribution for the inter-arrival time of market order placement by maximum likelihood estimation. Figure 3.5 displays the exponential fitting to the empirical distribution of the inter-arrival time of market order placement. Intuitively, there is a hump of the empirical distribution that cannot be well captured by the exponential distribution. Besides, the linear tail in the log-log plot also fails to be fitted by the exponential distribution.

3.4 Independence test

The classical assumptions of zero-intelligence models are the counting process of order flow follows homogeneous Poisson process. The independent inter-arrival time is then assumed during the modelling. We then conduct standard test to justify this assumption of zero-intelligence modelling for our data set.

To test the independence of the inter-arrival time, the Ljung–Box test is employed, which is supposed to test whether any of a group of autocorrelations of a time series are different from zero. The basic setting of Ljung-Box test is shown below:

- H_0: The data are independently distributed;
- H_a: The data are not independently distributed.

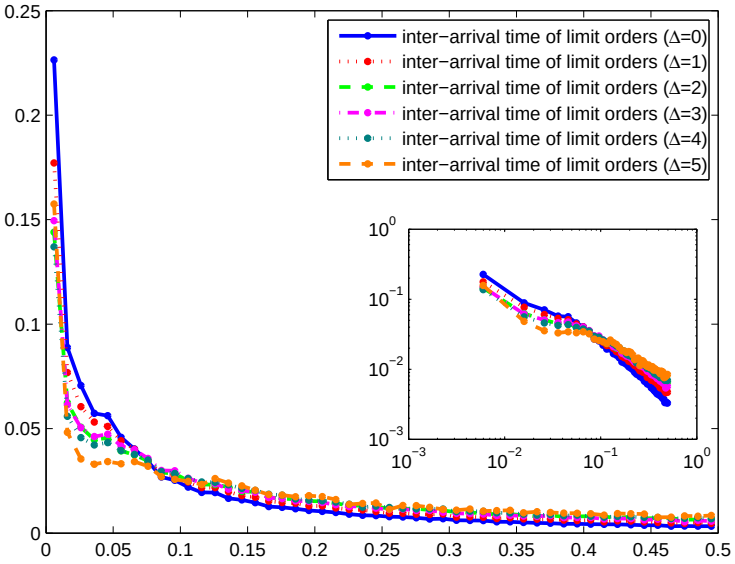

Figure 3.3: Empirical PDF of inter-arrival time of limit order placement

The test statistics is defined as

$$Q = n\,(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k} \tag{3.1}$$

where n is the sample size, h is the number of lags tested and $\hat{\rho}_k$ is the sample autocorrelation at lag k. This statistics follows a chi-square distribution χ_h^2.

The ACF and PACF of the inter-arrival time of limit order placement in our data set are shown in Figure 3.6. As an illustration, Figure 3.6 demonstrates the case of limit order placement on top of the book. Notice that the bound of ACF and PACF is ± 0.0027 in our data set, which implies that the ACF and PACF are both significant up to lags 20. The results of Ljung–Box test also indicates that all the flows of limit order placement at a relative price of i $(i = 0, 1, 2, ..., 5)$ fail the test with a p-value less than 0.001; while the flow of market order placement also fails the test with a p-value 0.001.

3.5 Other fittings

In the sequel, we focus on the limit order placement on top of the book (at a relative price of 0). For limit order placement at other relative price levels, we can follow the same procedures to study. We also focus on the market order placement.

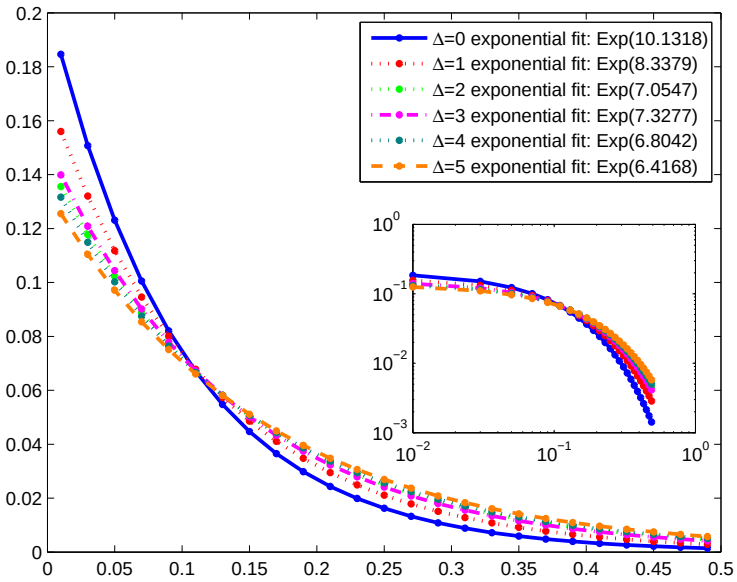

Figure 3.4: Exponential fit of inter-arrival time of limit order placement

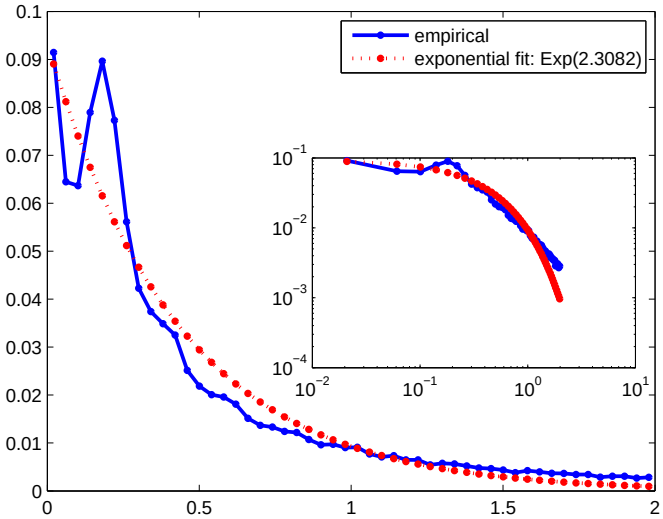

Figure 3.5: Exponential fit of inter-arrival time of market order placement

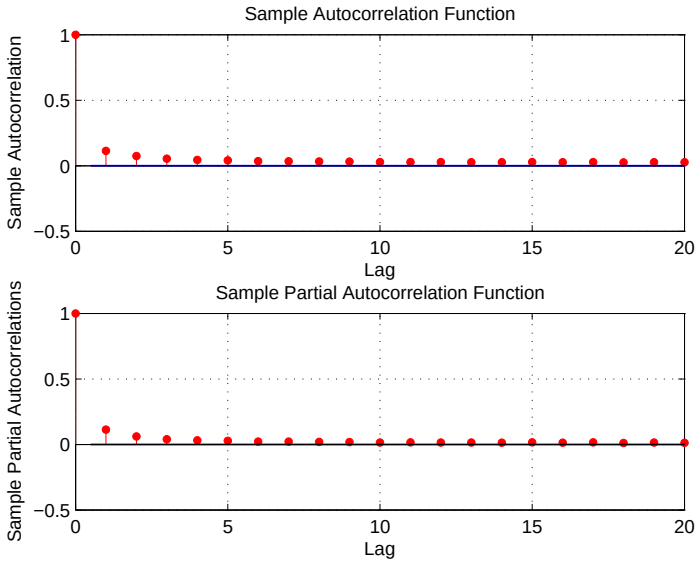

Figure 3.6: ACF and PACF of inter-arrival time of limit order placement

3.5.1 Gamma fitting

Since the exponential distribution is a particular case of a gamma distribution, we intend to employ a gamma distribution to fit the inter-arrival time of order placement. Note that there are two parameters in Gamma distribution $\Gamma(\gamma, \lambda)$, where γ captures the shape of distribution, and λ represents the arrival intensity (as exponential distribution does).

Figure 3.7 displays the gamma fitting estimated by the maximum likelihood method for the inter-arrival time of limit order placement on top of the book. The fitted parameters are 0.600 as the shape and 6.08 as the intensity. Clearly, the fitting performance of gamma distribution is remarkably better than exponential distribution, which suggests the shape parameter is helpful in capturing the distribution of the inter-arrival time of order flow. However, the linear tail is still unable to be captured by gamma distribution.

As to the market order placement, we follow the similar procedures to fit a gamma distribution. Figure 3.8 shows that the gamma fitting is very close to the exponential fitting, which implies that the shape parameter cannot improve the fitting performance. Since the estimated shape parameter is 0.928, which is fairly close to 1 (*i.e.* exponential distribution), it is natural that these two fittings demonstrate similar results.

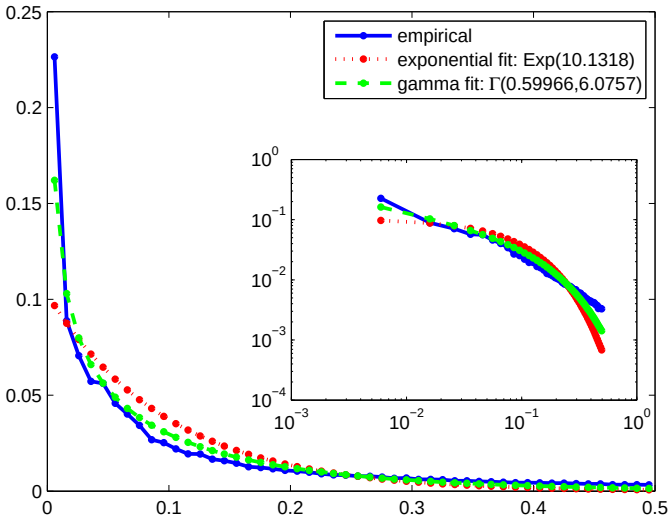

Figure 3.7: Gamma fit of inter-arrival time of limit order placement

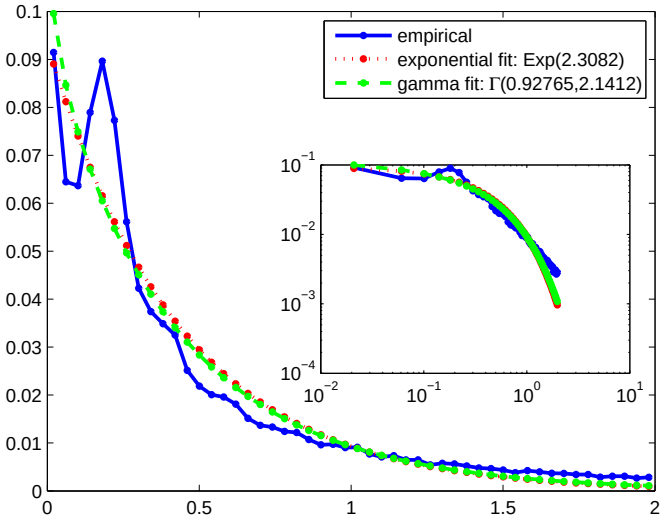

Figure 3.8: Gamma fit of inter-arrival time of market order placement

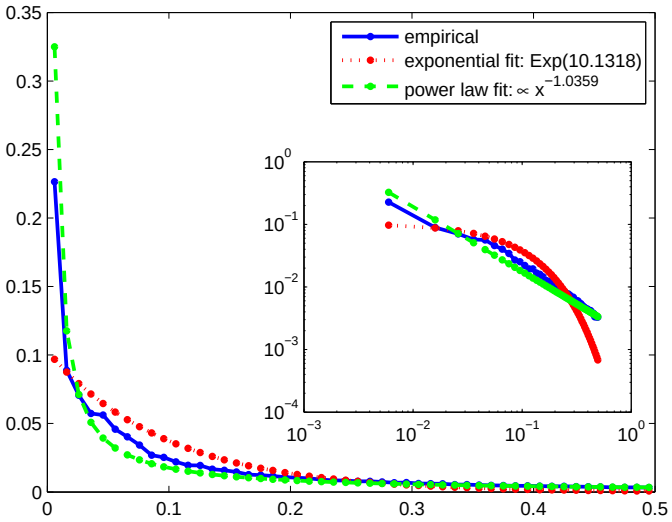

Figure 3.9: Power-law fit of inter-arrival time of limit order placement

3.5.2 Power-law fitting

Many empirical studies suggested a power-law fit for the inter-arrival time of order flows. Figure 3.9 demonstrates a power law fit for limit order placement on top of the book. We discovered an exponent of -1.04 for the inter-arrival time of limit order placement on top of the book. The decay speed around 0 seems to be over-exaggerated by power law. Nevertheless, a remarkable upside is that the tail can be well fitted by the power law.

Likewise, power-law is fitted to the inter-arrival time of market order placement. Figure 3.10 shows the fitting result for market order placement. The density around zero is remarkably exaggerated, which indicates that the power-law fitting is fairly poor for market order flow in our data set.

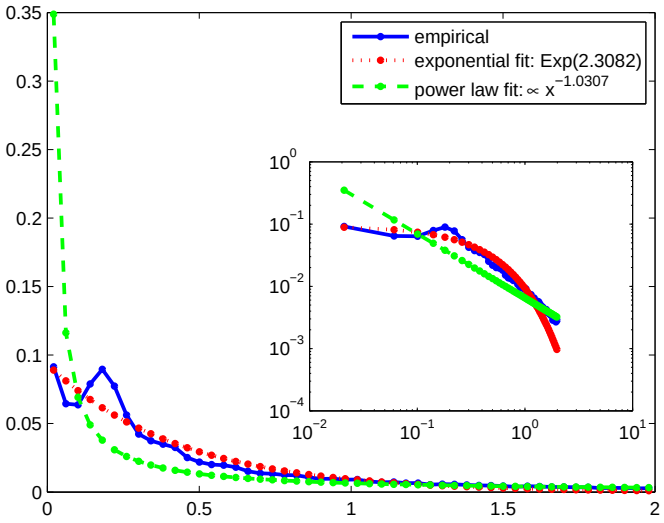

Figure 3.10: Power-law fit of inter-arrival time of market order placement

Chapter 4

Dependence of order flow

In Section 3, we have displayed several critical stylised facts of order flows in our data set. The inter-arrival time of limit order placement and market order placement were well studied, and fitted by exponential distribution, gamma distribution and power law separately. Nonetheless, all these distributions do not take the dependence of order flow into account. In this section, we will focus on the dependence of order flow. Hawkes process discussed in Section 2.2 then serves as a critical tool to capture the dependence.

4.1 Empirical facts

In the first place, we look into some empirical facts in terms of the dependence of order flow. The dependence of limit order placement on top of the book and market order placement are carefully studied. We focused on the flow of limit order placement on top of the book since this type of limit orders is most likely to be executed. Hence, the characteristics of limit order placement on top of the book can best represent the purchasing or selling pressure of the market. In the sequel, the flow of limit order placement always represents the limit orders placed on top of the book.

The *clustering effect* and *interplay effect* of these two order flows are expected. The clustering effect indicates that the arrival of a limit buy order (*resp.* limit sell order) will expedite the arrival of another limit buy order (*resp.* limit sell order). Likewise, the arrival of a market buy order (*resp.* market sell order) will expedite the arrival of another market buy order (*resp.* market sell order). From the empirical distribution of the inter-arrival time (Figure 3.3), the significant spike around zero for both limit order placement and market order placement implies a strong clustering effect.

Given the arrival of a limit buy order, we compare the number of limit buy order placement in t seconds beforehand versus the number of limit buy order placement in t seconds afterwards (called *LL effect*). Figure 4.1 then shows this comparison for three different time windows: 1 second, 5 seconds and 10 seconds. There is a clear uptrend for a time window

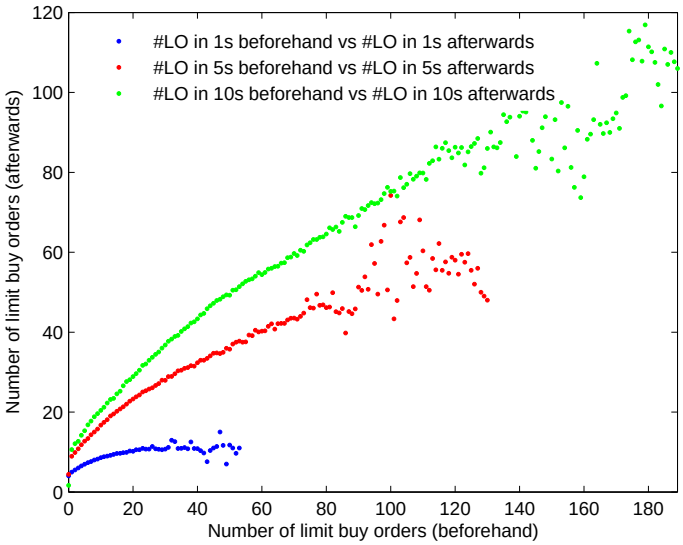

Figure 4.1: Clustering of order flow (LL effect)

of 5 seconds and 10 seconds, which implies a clustering effect: a small number of limit buy orders beforehand often triggers a small number of limit buy orders afterwards; while a large number of limit buy orders beforehand triggers a large number of limit buy orders afterwards. Note that similar clustering effect is also found for the limit sell order placement.

Figure 4.2 demonstrates the comparison between the number of market sell order placement in t seconds beforehand versus the number of market sell order placement in t seconds afterwards given the arrival of a market sell order (called *MM effect*), where t is chosen as 1 second, 5 seconds and 10 seconds respectively. Intuitively, MM effect is much weaker than LL effect for every compared time window. Nevertheless, for a time window of 10 seconds, MM effect is still significant. Similar study can be conducted to market buy order placement, which indicates similar results.

Apart from the clustering effect, the interplay effect of these two order flows are also expected in the market. It indicates that the arrival of a limit buy order will expedite the arrival of a market sell order. In turn, the arrival of a market sell order will expedite the arrival of a limit buy order.

We first compare the number of market sell order placement in t seconds beforehand versus the number of limit buy order placement in t seconds afterwards given the arrival of a limit order (called *ML effect*). Again three time windows were selected: 1 second, 5 seconds and 10 seconds. Figure 4.3 displays that ML effect is much weaker than LL effect

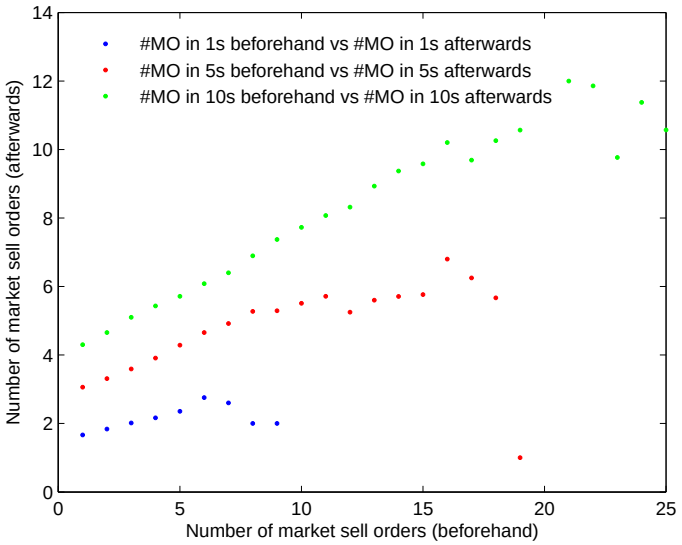

Figure 4.2: Clustering of order flow (MM effect)

(Figure 4.1). ML effect is merely significant for a time window of 10 seconds. The weak ML effect reveals that for limit order placement, the clustering effect is much stronger than the interplay effect.

We then compare the number of limit buy order placement in t seconds beforehand versus the number of market sell order placement in t seconds afterwards given the arrival of a market sell order (called *LM effect*). Figure 4.4 shows that LM effect is significant for the time window of 5 seconds and 10 seconds. LM effect indicates that when there is a clustering of limit order placement, the orders are more likely to be executed.

4.2 Estimation

Figure 4.1, 4.2, 4.3 and 4.4 show that there is a strong dependence in the flow of limit order placement and market order placement. The clustering effect and interplay effect can then be captured by self-excitation and mutual-excitation of Hawkes process. Equation 2.3 and Equation 2.4 are employed to simulate the inter-arrival time of these two order flows in our data set. Notice that self-excitation parameters α_{LL}, β_{LL} and α_{MM}, β_{MM} correspond to LL effect and MM effect in Section 4.1 while mutual-excitation parameters α_{ML}, β_{ML} and α_{LM}, β_{LM} correspond to ML effect and LM effect.

As an illustration, we focus on the limit buy order flow and market sell order flow. Similar procedures can then be taken on limit sell order flow and market buy order flow.

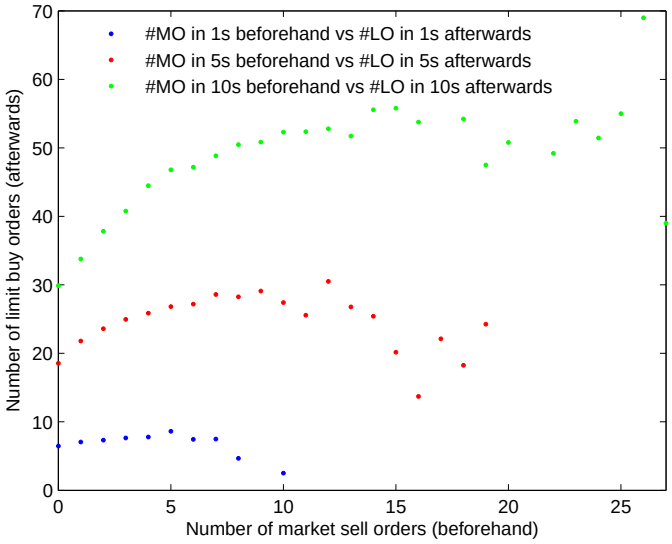

Figure 4.3: Interplay of order flow (ML effect)

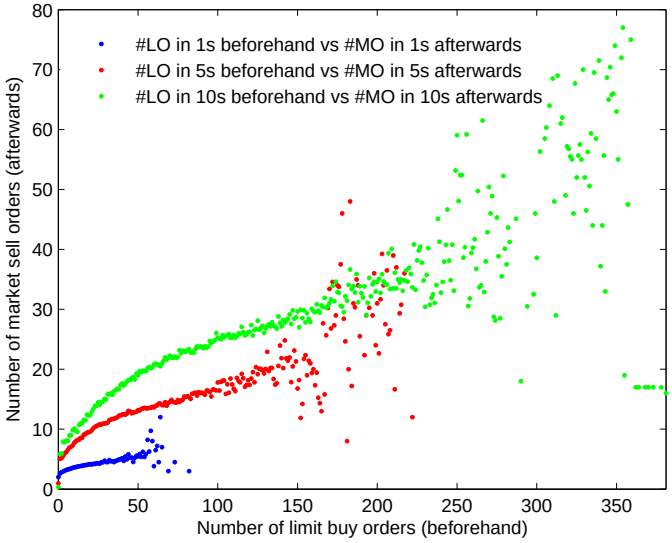

Figure 4.4: Interplay of order flow (LM effect)

The Hawkes parameters can be estimated by maximum likelihood method (see details in Section 2.3.1). To maximise the log-likelihood function, we use an active set algorithm so as to obtain the optimal parameters. The estimated parameters are shown in Table 4.1 and Table 4.2. The values in the brackets are estimated standard deviations. Notice that when incorporating mutual excitation effect, the intensity of self excitation effect will decrease, which implies that a part of intensity is captured by mutual excitation other than self excitation. Due to the time constraint, we will focus on the models with self-excitation effect in the sequel.

	λ_0^L	α_{LL}	β_{LL}	α_{ML}	β_{ML}
LL	0.67 (0.14)	3.92 (0.89)	28.57 (4.42)	-	-
LL+ML	0.56 (0.22)	3.78 (0.71)	31.57 (3.19)	0.98 (0.24)	17.23 (1.46)

Table 4.1: Hawkes parameters for limit buy order placement

4.3 Hawkes fitting

Following Algorithm 1, we are able to simulate the Hawkes processes with parameters fitted in Section 4.2. The illustration of Hawkes processes are shown in Figure 4.5 and Figure 4.6. The blue curve refers to the evolution of intensity and the red sticks represent the occurrences of events. Once there is an event occurring, there will immediately be an impact on the intensity. Obviously, the Hawkes process for limit order flow demonstrates a stronger intensity and clustering effect than the Hawkes process for market order flow.

Subsequently, we calculate the corresponding inter-arrival time from the simulated Hawkes processes. The fitting results are shown in Figure 4.7 and Figure 4.8. For limit buy order flow, Hawkes process captures the spike better around zero; however, it also fails to capture the tail as the exponential fit does. For market sell flow, Hawkes process fails to capture the spike around zero; however, it achieves a satisfactory fitting to the tail of empirical distribution.

	λ_0^M	α_{MM}	β_{MM}	α_{LM}	β_{LM}
MM	0.11 (0.03)	0.26 (0.05)	3.36 (0.09)	-	-
MM+LM	0.07 (0.01)	0.13 (0.03)	3.13 (0.32)	0.14 (0.06)	5.85 (0.78)

Table 4.2: Hawkes parameters for market sell order placement

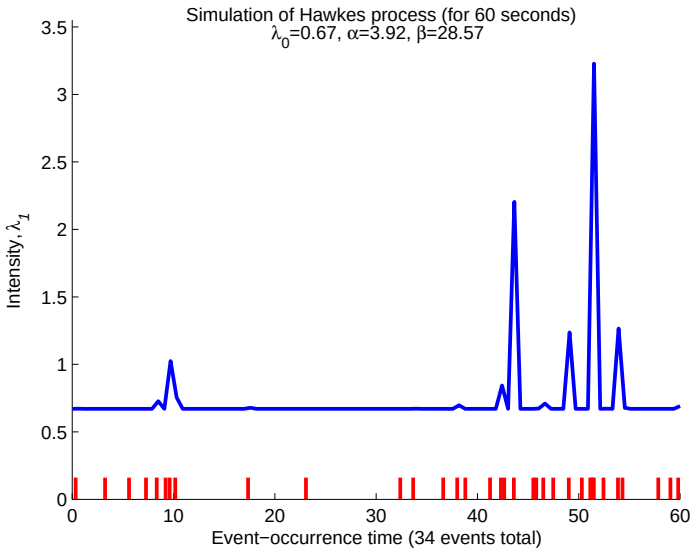

Figure 4.5: Simulation of Hawkes process (limit buy order flow)

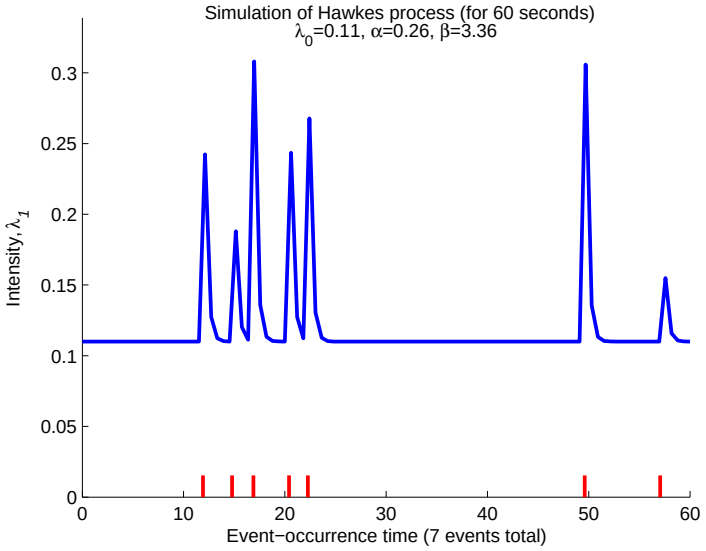

Figure 4.6: Simulation of Hawkes process (market sell order flow)

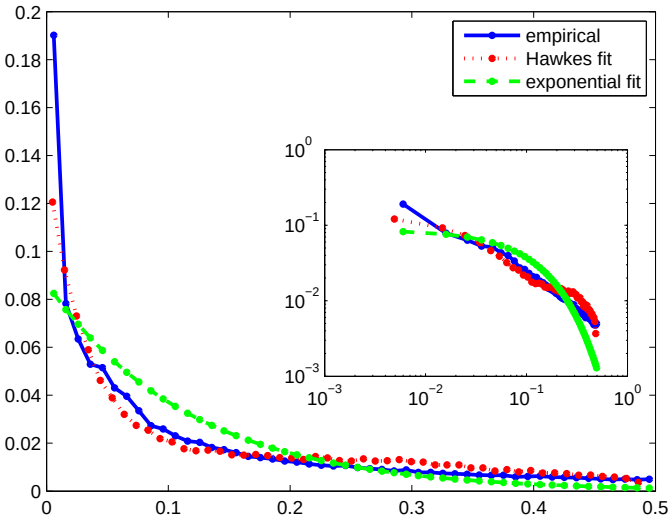

Figure 4.7: Hawkes fit of inter-arrival time of limit order placement

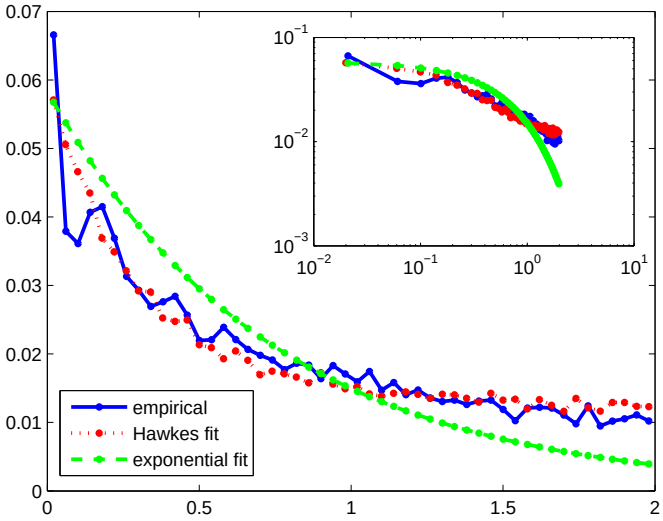

Figure 4.8: Hawkes fit of inter-arrival time of market order placement

4.4 Hawkes process with gamma distribution

In this section, we intend to generalise Hawkes process with a gamma distribution, since gamma distribution contains an additional shape parameter that may be conductive to improve the fitting performance. Note that a Gamma distribution $\Gamma(\gamma, \lambda)$ admits a probability density function

$$p(\gamma, \lambda) = \frac{\lambda^{\gamma}}{\Gamma(\gamma)} \tau^{\gamma-1} e^{-\lambda\tau} \tag{4.1}$$

And a univariate Hawkes process admits a stochastic density

$$\lambda_t = \lambda_0 + \int_0^t \alpha e^{-\beta(t-s)} dN_s \tag{4.2}$$

Hence, a Hawkes process with gamma distribution then admits a probability density function $p(\gamma, \lambda_t)$ where λ_t is defined in Equation (4.2).

4.4.1 Estimation

Apart from the triple parameters of a simple univariate Hawkes process $(\lambda_0, \alpha, \beta)$, there is an additional shape parameter γ for a Hawkes process with gamma distribution. The parameters can again be estimated by maximum likelihood method. Let t_i be the time stamp when i-th event occurs and $\tau_i = t_i - t_{i-1}$ be the inter-arrival time of $(i-1)$-th event and i-th event, then the likelihood function is

$$\mathcal{L}(\alpha, \beta, \lambda_0, \gamma | t_i) = \prod_i \left(\frac{\lambda_{t_i}^{\gamma}}{\Gamma(\gamma)} \tau_i^{\gamma-1} e^{-\lambda_{t_i}\tau_i} \right) \tag{4.3}$$

where $\lambda_{t_i} = \lambda_0 + \sum_{t_j < t_i} \alpha e^{-\beta(t_i - t_j)}$. We can then use a numerical method to maximise the log-likelihood function

$$l(\alpha, \beta, \lambda_0, \gamma | t_i) = \sum_i \left(\gamma \log \lambda_{t_i} - \log \Gamma(\gamma) + (\gamma - 1) \log \tau_i - \lambda_{t_i} \tau_i \right) \tag{4.4}$$

Table 4.3 shows the estimated parameters of a Hawkes process with gamma distribution for the limit buy order flow. The first row is the simple univariate Hawkes parameters estimated in Section 4.2, and the second row is the univariate Hawkes process with Gamma distribution. The shape parameter γ_M is significant in this case. In addition, the shape parameter also pushes down the estimated intensity in the process. Likewise, Table 4.4 shows the estimated parameters for the market sell order flow. It reveals that the shape

	λ_0^L	α_{LL}	β_{LL}	γ_L
LL effect (Exp)	0.67 (0.14)	3.92 (0.89)	28.57 (4.42)	1
LL effect (Gamma)	0.38 (0.08)	2.16 (0.26)	21.28 (3.40)	0.71 (0.04)

Table 4.3: Hawkes-Gamma parameters for limit buy order placement

	λ_0^M	α_{MM}	β_{MM}	γ_M
MM effect (Exp)	0.11 (0.03)	0.26 (0.05)	3.36 (0.09)	1
MM effect (Gamma)	0.06 (0.01)	0.16 (0.03)	3.11 (0.42)	0.92 (0.07)

Table 4.4: Hawkes-Gamma parameters for market sell order placement

parameter γ_M is not significant from 1, *i.e.* this Hawkes-Gamma fit will not be significantly different from a simple Hawkes fit.

4.4.2 Simulation

In order to evaluate the fitting performance of Hawkes-Gamma fit, a simulation algorithm is then required. Notice that this is a gamma distribution with a stochastic intensity, which is not easy to sample directly. We then turn to the rejection sampling from an exponential distribution as the candidate density function. We focus on the univariate Hawkes intensity (Equation (4.2)), then for $t_{i-1} \leq t_i$, we have

$$\begin{aligned}
\lambda_{t_i-} - \lambda_{t_{i-1}} &= \sum_{t_j<t_i} \alpha e^{-\beta(t_i-t_j)} - \sum_{t_j<t_{i-1}} \alpha e^{-\beta(t_{i-1}-t_j)} \\
&= \left(e^{-\beta(t_i-t_{i-1})} - 1\right) \sum_{t_j<t_{i-1}} \alpha e^{-\beta(t_{i-1}-t_j)} \\
&= \left(e^{-\beta(t_i-t_{i-1})} - 1\right) \left(\lambda_{t_{i-1}} - \lambda_0\right) \qquad (4.5)
\end{aligned}$$

then

$$\lambda_{t_i-} = \lambda_0 + e^{-\beta(t_i-t_{i-1})} \left(\lambda_{t_{i-1}} - \lambda_0\right) \qquad (4.6)$$

$$\lambda_{t_i} = \lambda_{t_i-} + \alpha \qquad (4.7)$$

where $\{t_i\}$ is a series of time stamps when there is an event occurring. Notice that Equation (4.6) and Equation (4.7) serves as a formula to update the intensity during the counting process. The acceptance-rejection algorithm can then be formulated as Algorithm 2.

Algorithm 2 Acceptance-rejection algorithm

1. **Sampling from the candidate distribution**: Generate $\tau_i \sim Gamma\left(\gamma, \lambda_{t_{i-1}}\right)$.
2. **Update of intensity before jump**: Calculate λ_{t_i-} from Equation (4.6);
3. **Calculation of acceptance probability**: Calculate the acceptance probability

$$P^* = \frac{p\left(\gamma, \lambda_{t_i-}\right)}{p\left(\gamma, \lambda_{t_{i-1}}\right)} = \left(\frac{\lambda_{t_i-}}{\lambda_{t_{i-1}}}\right)^{\gamma} e^{-\left(\lambda_{t_i-}-\lambda_{t_{i-1}}\right)\tau_i} \tag{4.8}$$

   where $p\left(\gamma, \lambda\right)$ is the probability density function of a gamma distribution. If $P^* > 1$, then go back to 1;

4. **Acceptance/ Rejection Rule**: Generate $U \sim \mathcal{U}\left[0, 1\right]$.
   (a) If $U \leq P^*$, then accept τ_i as a sample of Hawkes-Gamma and update λ_{t_i} by Equation (4.7). Afterwards go back to 1;
   (b) If $U > P^*$, then reject τ_i . Afterwards go back to 1.

Notice that the acceptance probability (Equation 4.8) is bounded by 1, since

$$P^* = \left(\frac{\lambda_{t_i-}}{\lambda_{t_{i-1}}}\right)^{\gamma} e^{-\left(\lambda_{t_i-}-\lambda_{t_{i-1}}\right)\tau_i} \leq \left(\frac{\lambda_{t_i-}}{\lambda_{t_{i-1}}}\right)^{\gamma} \leq 1 \tag{4.9}$$

Following Algorithm 2, we simulate the Hawkes-Gamma model and fit for both limit order placement and market order placement. Figure 4.9 demonstrates that a Hawkes-Gamma fitting captures better around 0 and tails than a simple Hawkes fitting, which indicates that Hawkes-Gamma is a better choice to model limit order placement in our data set. Figure 4.10 displays that the difference between a Hawkes-Gamma fitting and a simple Hawkes fitting is not significant, which corresponds to the insignificance of γ_M from 1.

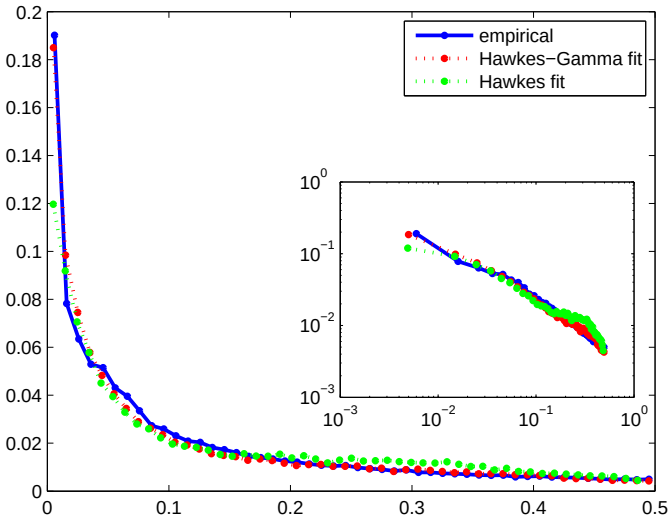

Figure 4.9: Hawkes-Gamma fit of inter-arrival time of limit order placement

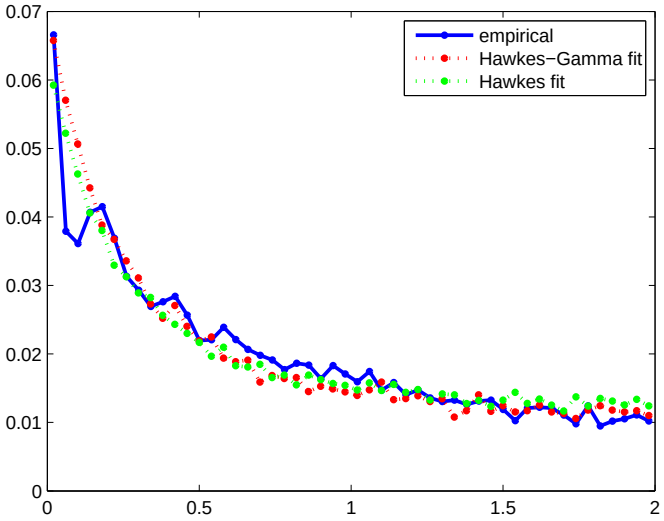

Figure 4.10: Hawkes-Gamma fit of inter-arrival time of market order placement

Chapter 5

Conclusion

In this thesis, we studied the features of limit order flow and market order flow. The inter-arrival time of orders are fitted by exponential distribution, gamma distribution, power-law, simple Hawkes process and Hawkes-Gamma process.

Exponential distribution is the standard assumption of zero-intelligence models; however it achieves the worst fitting performance in our data set. Gamma distribution contains an extra parameter of exponential distribution, which can capture the shape of the distribution; however it fails to fit the tail of the empirical distribution. Power-law still fits very well to the tail of the empirical distribution; however it over-exaggerates the decay speed around zero.

When incorporating the dependence into the counting process of order flow, Hawkes process is employed to represent the clustering effect and the interplay effect. We studied the fitting results with the clustering effect, which reveal much better than exponential distribution. In addition, we combined Hawkes process with gamma distribution, which can capture both the dependence and the shape of the empirical distribution. This new approach provides a powerful tool in modelling the counting process of order flow. Due to the time constraint, only the self-excitation Hawkes-Gamma model was studied. As a future work, Hawkes-Gamma model with mutual-excitation can also be studied, which may offer an even better model.

Bibliography

[1] Anand, A., Chakravarty, S. and Martell, T., 2005. Empirical evidence on the evolution of liquidity: choice of market versus limit orders by informed and uninformed traders.

[2] Biais, B., Hilliton, P. and Spatt, C., 1995. An empirical analysis of the limit order book and the order flow in the Paris Bourse.

[3] Bouchaud, J-P., Mezard, M. and Potters, M., 2002. Statistical properties of stock order books: Empirical results and models.

[4] Bouchaud, J-P, Farmer, J. D., and Lillo, F., 2009. How markets slowly digest changes in supply and demand.

[5] Bowsher, C. G., 2003. Modelling security market events in continuous time: intensity based, multivariate point process models

[6] Bremaud, P. and Massoulie, L., 1996. Stability of nonlinear Hawkes processes.

[7] Cont, R., Stoikov S. and Talreja, R., 2008. A stochastic model for order book dynamics.

[8] Farmer, J. D., Patelli, P. and Zovko, I., 2005. The predictive power of zero intelligence in financial markets.

[9] Gould, M., Porter, M., Williams, S., McDonald, M., Fenn, D. & Howison, S., 2011. Limit order book.

[10] Harris, L. and Hasbrouck J., 1996. Market vs. limit orders: the SuperDOT evidence on order submission strategy.

[11] Harris, L., 2002. Trading and exchanges: market microstructure for practitioners.

[12] Hawkes, A., 1971. Spectra of some self-exciting and mutually exciting point processes.

[13] Kirilenko, A., Kyle, A., Samadi, M. and Tuzun, T., 2011. The flash crash: the impact of high-frequency trading on an electronic market.

[14] Lehmann, E. L. and Casella, G., 1998. Theory of point estimation (second edition).

[15] Nocedal, J. and Wright, S. J., 2006. Numerical optimization (second edition).

[16] Ogata, Y., 1981. On Lewis' simulation method for point processes.

[17] Ozaki, T., 1979. Maximum likelihood estimation of Hawkes' self-exciting point processes.

[18] Parlour, C. A., and Seppi, D. J., 2007. Limit order markets: a survey.

[19] Smith, E., Farmer, J. D., Gillemot, L. and Krishnamurthy S., 2002. Statistical theory of the continuous double auction.

[20] Stuart, A., Ord, K. and Arnold, S., 1999. Classical inference and the linear model.

[21] Toke, I. M. and Pomponio, F., 2012. Modelling trades-through in a limit order book using hawkes processes.

[22] Zovko, I. and Farmer, J. D., 2002. The power of patience: a behavioral regularity in limit order placement

Appendix: Matlab codes

A. Pre-processing data

A1. Codes for event identification

This function deals with the data file and returns a matrix of market events, where each row contains timestamp, milli-second, best bid, best ask, event type, order size, order price and order direction.

```
function [event,tradeVol] = Identification(instr,date)

filepath = 'D:\XZheng\Dissertation\Data\';
filename = [instr date '.mat'];
load([filepath filename]);
book(book(:,1)<70000 | book(:,1)>190000,:) = [];     % 07:00-19:00
book(:,2) = round(book(:,2)./1000000);               % millisec

% outlier removing
avg = mean(book(:,5));
sd = std(book(:,5));
ub = avg + 30*sd;
lb = avg - 30*sd;
book(book(:,5)>ub | book(:,5)<lb,:) = [];

n = 1;
event = zeros(size(book,1),8);
tradeVol = 0;

for i = 7:(size(book,1)-6)
    delta = book(i,5:end)-book(i-1,5:end);

    idxBID = find(delta(1:5) ~= 0);
    idxASK = find(delta(6:10) ~= 0);
    idxBSIZ = find(delta(11:15) ~= 0);
    idxASIZ = find(delta(16:20) ~= 0);

    if isempty([idxBID idxASK idxBSIZ idxASIZ])
        continue
    end

    if isempty(idxBID), idxBID = 0; end
    if isempty(idxASK), idxASK = 0; end
    if isempty(idxBSIZ), idxBSIZ = 0; end
    if isempty(idxASIZ), idxASIZ = 0; end

    status = book(i-1,[1 2 5 10]);
```

```
38
39      % Bid side
40      if idxBID(1)~=0 || idxBSIZ(1)~=0
41          d = 1;                          % direction
42          % Level 1
43          if idxBID(1)==1 || idxBSIZ(1)==1
44              if delta(1) < 0
45
46                  p = book(i-1,5);     % order price
47                  w = book(i-1,15);    % order size
48                  rangeVol = unique(book((i-6):(i+6),4));
49                  isExecuted = (size(rangeVol,1)>1) & ...
50                      ~isempty(find(rangeVol == w+tradeVol, 1));
51                  if ~isExecuted
52                      type = 3;            % event type
53                      event(n,:) = [status type w p d];
54                      n = n + 1;
55                  else
56                      type = 5;
57                      tradeVol = tradeVol + w;
58                      event(n,:) = [status type w p d];
59                      n = n + 1;
60                  end
61
62                  p = book(i,5);
63                  w = book(i,15);
64                  idxPrev = find(book(i-1,5:9)==p, 1);
65                  if isempty(idxPrev)
66                      type = 1;
67                      event(n,:) = [status type w p d];
68                      n = n + 1;
69                  else
70                      wPrev = book(i-1,idxPrev+14);
71                      if w > wPrev
72                          w = w-wPrev;
73                          type = 1;
74                          event(n,:) = [status type w p d];
75                          n = n + 1;
76                      elseif w < wPrev
77                          w = wPrev-w;
78                          isExecuted = (size(rangeVol,1)>1) & ...
79                              ~isempty(find(rangeVol == w+tradeVol, 1));
80                          if ~isExecuted
81                              type = 2;
82                              event(n,:) = [status type w p d];
83                              n = n + 1;
84                          else
```

```
85                              type = 4;
86                              tradeVol = tradeVol + w;
87                              event(n,:) = [status type w p d];
88                              n = n + 1;
89                          end
90                      end
91                  end
92
93              elseif delta(1) > 0
94                  p = book(i,5);
95                  w = book(i,15);
96                  type = 1;
97                  event(n,:) = [status type w p d];
98                  n = n + 1;
99
100                 p = book(i-1,5);
101                 w = book(i-1,15);
102                 idxNext = find(book(i,5:9)==p, 1);
103                 if isempty(idxNext)
104                     type = 3;
105                     event(n,:) = [status type w p d];
106                     n = n + 1;
107                 else
108                     wNext = book(i, idxNext+14);
109                     if w > wNext
110                         w = w-wNext;
111                         rangeVol = unique(book((i-6):(i+6),4));
112                         isExecuted = (size(rangeVol,1)>1) & ...
113                             ~isempty(find(rangeVol == w+tradeVol, 1));
114                         if ~isExecuted
115                             type = 2;
116                             event(n,:) = [status type w p d];
117                             n = n + 1;
118                         else
119                             type = 4;
120                             tradeVol = tradeVol + w;
121                             event(n,:) = [status type w p d];
122                             n = n + 1;
123                         end
124                     elseif w < wNext
125                         w = wNext-w;
126                         type = 1;
127                         event(n,:) = [status type w p d];
128                         n = n + 1;
129                     end
130                 end
131
```

```
132             else                % delta(1) == 0
133                 p = book(i,5);
134                 w = book(i,15);
135                 wPrev = book(i-1,15);
136                 if w > wPrev
137                     w = w-wPrev;
138                     type = 1;
139                     event(n,:) = [status type w p d];
140                     n = n + 1;
141                 elseif w < wPrev
142                     w = wPrev-w;
143                     rangeVol = unique(book((i-6):(i+6),4));
144                     isExecuted = (size(rangeVol,1)>1) & ...
145                         ~isempty(find(rangeVol == w+tradeVol, 1));
146                     if ~isExecuted
147                         type = 2;
148                         event(n,:) = [status type w p d];
149                         n = n + 1;
150                     else
151                         type = 4;
152                         tradeVol = tradeVol + w;
153                         event(n,:) = [status type w p d];
154                         n = n + 1;
155                     end
156                 end
157             end
158         end
159
160         % Level 2~5
161         for lv = 2:5
162             if idxBID(1)==lv || idxBSIZ(1)==lv
163                 if delta(lv) < 0
164                     p = book(i-1,lv+4);
165                     w = book(i-1,lv+14);
166                     type = 3;
167                     event(n,:) = [status type w p d];
168                     n = n + 1;
169
170                 elseif delta(lv) > 0
171                     p = book(i,lv+4);
172                     w = book(i,lv+14);
173                     type = 1;
174                     event(n,:) = [status type w p d];
175                     n = n + 1;
176
177                 else        % delta(lv) = 0
178                     p = book(i,lv+4);
```

```
179                     w = book(i,lv+14);
180                     wPrev = book(i-1,lv+14);
181                     if w > wPrev
182                         w = w-wPrev;
183                         type = 1;
184                         event(n,:) = [status type w p d];
185                         n = n + 1;
186                     elseif w < wPrev
187                         w = wPrev-w;
188                         type = 2;
189                         event(n,:) = [status type w p d];
190                         n = n + 1;
191                     end
192                 end
193             end
194         end
195     end
196
197     % Ask side
198     if idxASK(1)~=0 || idxASIZ(1)~=0
199         d = -1;                     % direction
200         % Level 1
201         if idxASK(1)==1 || idxASIZ(1)==1
202             if delta(6) > 0
203                 p = book(i-1,10);
204                 w = book(i-1,20);
205                 rangeVol = unique(book((i-6):(i+6),4));
206                 isExecuted = (size(rangeVol,1)>1) & ...
207                     ~isempty(find(rangeVol == w+tradeVol, 1));
208                 if ~isExecuted
209                     type = 3;
210                     event(n,:) = [status type w p d];
211                     n = n + 1;
212                 else
213                     type = 5;
214                     tradeVol = tradeVol + w;
215                     event(n,:) = [status type w p d];
216                     n = n + 1;
217                 end
218
219                 p = book(i,10);
220                 w = book(i,20);
221                 idxPrev = find(book(i-1,10:14)==p, 1);
222                 if isempty(idxPrev)
223                     type = 1;
224                     event(n,:) = [status type w p d];
225                     n = n + 1;
```

```
226                 else
227                     wPrev = book(i-1,idxPrev+19);
228                     if w > wPrev
229                         w = w-wPrev;
230                         type = 1;
231                         event(n,:) = [status type w p d];
232                         n = n + 1;
233                     elseif w < wPrev
234                         w = wPrev-w;
235                         isExecuted = (size(rangeVol,1)>1) & ...
236                             ~isempty(find(rangeVol == w+tradeVol, 1));
237                         if ~isExecuted
238                             type = 2;
239                             event(n,:) = [status type w p d];
240                             n = n + 1;
241                         else
242                             type = 4;
243                             tradeVol = tradeVol + w;
244                             event(n,:) = [status type w p d];
245                             n = n + 1;
246                         end
247                     end
248                 end
249
250             elseif delta(6) < 0
251                 p = book(i,10);
252                 w = book(i,20);
253                 type = 1;
254                 event(n,:) = [status type w p d];
255                 n = n + 1;
256
257                 p = book(i-1,10);
258                 w = book(i-1,20);
259                 idxNext = find(book(i,10:14)==p, 1);
260                 if isempty(idxNext)
261                     type = 3;
262                     event(n,:) = [status type w p d];
263                     n = n + 1;
264                 else
265                     wNext = book(i, idxNext+19);
266                     if w > wNext
267                         w = w-wNext;
268                         rangeVol = unique(book((i-6):(i+6),4));
269                         isExecuted = (size(rangeVol,1)>1) & ...
270                             ~isempty(find(rangeVol == w+tradeVol, 1));
271                         if ~isExecuted
272                             type = 2;
```

```
273                             event(n,:) = [status type w p d];
274                             n = n + 1;
275                         else
276                             type = 4;
277                             tradeVol = tradeVol + w;
278                             event(n,:) = [status type w p d];
279                             n = n + 1;
280                         end
281                     elseif w < wNext
282                         w = wNext-w;
283                         type = 1;
284                         event(n,:) = [status type w p d];
285                         n = n + 1;
286                     end
287                 end
288 
289             else                    % delta(6) == 0
290                 p = book(i,10);
291                 w = book(i,20);
292                 wPrev = book(i-1,20);
293                 if w > wPrev
294                     w = w-wPrev;
295                     type = 1;
296                     event(n,:) = [status type w p d];
297                     n = n + 1;
298                 elseif w < wPrev
299                     w = wPrev-w;
300                     rangeVol = unique(book((i-6):(i+6),4));
301                     isExecuted = (size(rangeVol,1)>1) & ...
302                         ~isempty(find(rangeVol == w+tradeVol, 1));
303                     if ~isExecuted
304                         type = 2;
305                         event(n,:) = [status type w p d];
306                         n = n + 1;
307                     else
308                         type = 4;
309                         tradeVol = tradeVol + w;
310                         event(n,:) = [status type w p d];
311                         n = n + 1;
312                     end
313                 end
314             end
315         end
316 
317         % Level 2~5
318         for lv = 2:5
319             if idxASK(1)==lv || idxASIZ(1)==lv
```

```
320                 if delta(lv+5) < 0
321                     p = book(i-1,lv+9);
322                     w = book(i-1,lv+19);
323                     type = 3;
324                     event(n,:) = [status type w p d];
325                     n = n + 1;
326
327                 elseif delta(lv+5) > 0
328                     p = book(i,lv+9);
329                     w = book(i,lv+19);
330                     type = 1;
331                     event(n,:) = [status type w p d];
332                     n = n + 1;
333
334                 else        % delta(lv+5) = 0
335                     p = book(i,lv+9);
336                     w = book(i,lv+19);
337                     wPrev = book(i-1,lv+19);
338                     if w > wPrev
339                         w = w-wPrev;
340                         type = 1;
341                         event(n,:) = [status type w p d];
342                         n = n + 1;
343                     elseif w < wPrev
344                         w = wPrev-w;
345                         type = 2;
346                         event(n,:) = [status type w p d];
347                         n = n + 1;
348                     end
349                 end
350             end
351         end
352     end
353
354     if book(i,4) == 0
355         continue
356     end
357
358     % execution against hidden order
359     if tradeVol+book(i,3)==book(i,4)
360         type = 6;       % verified hidden order
361         tradeVol = book(i,4);
362         w1 = book(i,4)-tradeVol;
363         event(n,:) = [status type w1 0 0];
364         n = n + 1;
365     elseif tradeVol+book(i,3)<book(i,4)
366         type = 7;       % unverified hidden order
```

```
367         tradeVol = book(i,4);
368         w1 = book(i,4)-tradeVol;
369         event(n,:) = [status type w1 0 0];
370         n = n + 1;
371     end
372
373     disp(['Line ' num2str(i) ' is processed.']);
374 end
375
376 event(event(:,1) == 0,:) = [];
377
378 end
```

B. Codes for Hawkes parameter estimation

B1. Univariate Hawkes

This function calculates the negative log-likelihood function of a univariate Hawkes.

```
1  function f = neg_loglik_uni(para,t)
2
3  lambda0 = para(1);
4  alpha = para(2);
5  beta = para(3);
6  t = sort(t);
7  n = size(t,1);
8  r = zeros(n,1);
9  for i = 2:n
10     r(i) = exp(-beta*(t(i)-t(i-1)))*(1+r(i-1));
11 end
12 f = t(n) - lambda0*t(n) - sum(alpha/beta.*(1-exp(-beta.*(t(n)-t)))) ...
13     + sum(log(lambda0 + alpha.*r));
14 f = -f;
15
16 end
```

B2. Bivariate Hawkes

These two functions calculate the negative log-likelihood function of a bivariate Hawkes.

```
1  function f = neg_loglik_bi_partial(para,t1,t2)
2  % based on intensity of t1.
3
4  lambda0 = para(1);
5  alpha1 = para(2);        % self-excitation
6  beta1 = para(3);
```

```
alpha2 = para(4);        % mutual-excitation
beta2 = para(5);

t1 = sort(t1);
t2 = sort(t2);
n1 = size(t1,1);
r1 = zeros(n1,1);
r2 = zeros(n1,1);
for i = 2:n1
    r1(i) = exp(-beta1*(t1(i)-t1(i-1)))*(1+r1(i-1));
end
for i = 2:n1
    r2(i) = exp(-beta2*(t1(i)-t1(i-1)))*r2(i-1);
    temp = t2(t2>=t1(i-1) & t2<t1(i),:);
    if isempty(temp)
        continue
    end
    for j = 1:size(temp,1)
       r2(i) = r2(i) + exp(-beta2*(t1(i)-temp(j)));
    end
end

f = t1(n1) - lambda0*t1(n1) - sum(alpha1/beta1.*(1-exp(-beta1.*(t1(n1)-t1)
  ))) ...
    - sum(alpha2/beta2.*(1-exp(-beta2.*(t1(n1)-t1)))) ...
    + sum(log(lambda0 + alpha1.*r1 + alpha2.*r2));
f = -f;

end
```

```
function f = neg_loglik_bi(para,t1,t2)

f1 = neg_loglik_bi_partial(para(1:5),t1,t2);
f2 = neg_loglik_bi_partial(para(6:10),t2,t1);
f = -(f1+f2);

end
```

B3. Estimation of parameters

This file estimates the parameters of Hawkes process.

```
[t1,t2] = HawkesData(instr,num2str(date));

para0 = [3;1;2];
[para,fval,~,~,grad,hessian] = fminunc(@(para) neg_loglik_uni(para,t),
  para0);
```

```
sd = sqrt(diag(inv(hessian)));

para0 = [3;1;2;1;2;3;1;2;1;2];
[para,fval,~,~,grad,hessian] = fminunc(@(para) neg_loglik_bi(para,t1,t2),
  para0);
sd = sqrt(diag(inv(hessian)));
```

C. Codes for Hawkes-Gamma

C1. Code for estimation

This function calculates the negative log-likelihood function for a Hawkes-Gamma process.

```
function f = neg_loglik_gamma_hawkes(para,t)

lambda0 = para(1);
alpha = para(2);
beta = para(3);
gam = para(4);
t = sort(t);
n = size(t,1);
tau = diff(t);
f = -n*log(gamma(gam)) + (gam-1)*sum(log(tau(tau~=0)),1);
for i = 2:n
    lambda = flambda(lambda0,alpha,beta,t(i),t);
    f = f + gam*log(lambda) - lambda*tau(i-1);
end
f = -f;

end
```

```
function lambda = flambda(lambda0,alpha,beta,ti,t)

temp = t(t<ti);
lambda = lambda0 + sum(alpha.*exp(-beta.*(ti-temp)),1);

end
```

C2. Code for simulation

This function simulates a Hawkes process with gamma distribution. Note that when gam=1, it can also be used to simulate a simple Hawkes process.

```
function x = HawkesGammaSimul(n,lambda0,alpha,beta,gam)

x = zeros(n,1);
```

```
i = 1;
lambda = lambda0;
while i ~= (n+1)
    tau = gamrnd(gam,1/lambda);
    lambda_temp = lambda0 + exp(-beta*tau)*(lambda-lambda0);
    P = (lambda_temp/lambda)^gam*exp(-(lambda_temp-lambda)*tau);
    if P > 1
        continue
    end
    U = rand;
    if U <= P
       x(i) = tau;
        i = i + 1;
        lambda = lambda_temp + alpha;
    end
end

end
```