Statistical Arbitrage in High Frequency Trading Based on Limit Order Book Dynamics

Murat Ahmed, Anwei Chai, Xiaowei Ding, Yunjiang Jiang, Yunting Sun

June 11, 2009

1 Introduction

Classic asset pricing theory assumes prices will eventually adjust to and reflect the fair value, the route and speed of transition is not specified. Market Microstructure studies how prices adjust to reflect new information. Recent years have seen the widely available high frequency data enabled by the rapid advance in information technology. Using high frequency data, it's interesting to study the roles played by the informed traders and noise traders and how the prices are adjusted to reflect information flow. It's also interesting to study whether returns are more predictable in the high frequency setting and whether one could exploit limit order book dynamics in trading.

Broadly speaking, the traditional approach to statistical arbitrage is through attempting to bet on the temporal convergence and divergence of price movements of pairs and baskets of assets, using statistical methods. A more academic definition of statistical arbitrage is to spread the risk among thousands to millions of trades in very short holding time, hoping to gain profit in expectation through the law of large numbers. Following this line, recently, a model based approach has been proposed by Rama Cont and coauthors [1], based on a simple birth-death markov chain model. After the model is calibrated to the order book data, various types of odds can be computed. For example, if a trader could estimate the probability of mid-price uptick movement conditional on the current orderbook status and if the odds are in his/her favor, the trader could submit an order to capitalize the odds. When the trade is carefully executed with a judicious stop-loss, the trader should be able to make profit in expectation.

In this project, we adopted a data-driven approach. We first built an "simulated" exchange order matching engine which allows us to reconstruct the orderbook. Therefore, in theory, we've built an exchange system which allows us to not only back-test our trading strategies but also evaluate the price impacts of trading. And we then implemented, calibrated and tested the Rama Cont model on both simulated data and real data. We also implemented, calibrated and tested an extended model. Based on these models and based on the orderbook dynamics, we explored a few high frequency trading strategies.

In Section 2, we discuss the "simulated" exchange order matching engine. In Section 3, we present a few statistical observations and stylized facts about data. In Section 4, we review the Rama Cont model of order book dynamics and extend the Rama Cont model. By

Table 1: Data Snippet from Intel on March 09, 2009

inputsymbol	timestamp	nts	sq	mm	bp	bs	ap	as
intc.o2	2009-03-09 04:07:39.017404079	1236586059.0	1070	ARCX	12.32	7	12.56	5
intc.o2	2009-03-09 04:07:55.587754011	1236586075.0	1081	ARCX	,	,	12.56	5

incorporating the covariates and using the Cox Proportional Hazards Model, we would like to capture some of the stylized facts of high frequency data such as co-movements among assets and the clustering effect. In Section 5, we present the results of our trading strategy experiments and finally, Section 6 concludes the report.

2 The Data and Our Order Matching Engine

The data we have on hand are about 10 days of selected stocks, e.g. Intel, QQQQ, etc., from March 9, 2009 to March 20, 2009, from about 4:00am to 8:00pm everyday.

Each data file comes in the format of CSV and contains millions to tens of millions of lines. The following is a snippet.

All the fields are quite self-explanatory, where "bp", "bs", "ap" and "as" denote the bid price, bid size, ask price, and ask size, respectively.

To interpret the data, we made the following assumptions:

- These are authentic Level 2 order arrival data in the raw message format. They are not the Level 1 data entailing best bids and best asks.
- These are continuous streaming data, instead of snapshots sampled at fixed intervals.
- These are raw data at the atomic level, therefore there are only three types of orders that are relevant: Market Orders, Limit Orders and Cancellation Orders. At higher investor levels, there are more fancy orders such as the stop-loss, stop-limit-loss, good-till-cancel orders, etc.
- Market Orders are defined to be orders that are immediately transacted.
- Limit Orders are defined to be orders that are not immediately transacted, therefore consumes liquidity.
- The message with commas defines the cancelation order. Two commas on the bid side cancel the previous bid quote. The two numbers on the ask side are the new arrival of quotes. The four-comma rows are cancelation orders on both sides.
- Cancelation order should cancel the previous orders from the same market maker, if that order was not filled; if the previous order was already filled, then the cancelation order is not effective and should be considered as void.

- When there are consecutive cancelation orders, we will recursively cancel the previous orders from the same market maker, one by one, in back order of time, unless the order-to-be-canceled has already been filled, in which case the corresponding cancelation order is considered as void and ineffective.

- Time Precedence: Orders arrive earlier have priority.

- Order Type Precedence: Market Order $\succeq$ Limit Order $\succeq$ Cancelation Order.

- The messages are mostly in order of arrival time. However we did observe out-of-order messages. In that case, we just assume that the data-reporting facility had made errors and we correct the errors by re-ordering the messages in order of time by using stable sort.

- Each message contains orders from both the bid and ask sides, this is because the market makers want to save bandwidth. In the cancellation message, one pair of commas are often bundled with limit or market orders from the opposite side. Again, our reasoning is that the market makers don't want send one message for each single order, and their liquidity is sufficient so that they could pool orders from both sides into one single message.

- We don't discard the line with commas because we think with the advance of electronic trading, the cancellation orders should consume a significant fraction of the overall orders. The cancellation orders at the atomic level come from the modification orders at the high level, which are very common in electronic trading.

- When executing a trade, the market order is first matched with the best opposite quote, and then the residue is matched with the second best opposite quote, so on and so forth.

- The data file is self-contained so we don't need any extra information such as times & sales data. Indeed, the data in this format actually contains much more information than the typical Level 2 data.

- The last trade price is defined by the execution of market orders. If a large market order is matched to multiple price levels of the opposite best quotes, the last trade price is recorded accordingly.

To reconstruct the order book dynamics, we built an order matching engine, which does the following:

- For each message, classify the bid order into market order, limit order and cancellation order, and do the same for the ask side.

- For market order, we just fill it immediately.

- For limit order, we just insert them into the orderbooks.

- For cancellation order, we search the separately maintained market-maker book and look for previous orders from the same market maker that are not yet filled. If that particular order has been already filled, we have to ignore the cancellation order. This reflects the fact that a trader might not be fast enough when he regrets about a previously submitted trade.

- Update the last trade price and other times & sales trade statistics.

- For later study of trading strategies, our order matching engine allows us, the Stanford 444 A.C.D.J.S. team, to submit our own orders and quotes, in an arbitrary manner, and all the orders will be interleaved and interact with each other.

Therefore, in theory, we've built an exchange system which allows us to not only back-test our trading strategies but also evaluate the price impacts of trading.

Based on our order matching rules, we've reconstructed the orderbook for real data. Using Intel on March 09, 2009 as an example, we have recorded from our data, the executed trade to be 96513473 shares, whereas Yahoo Finance record 87044700 shares. The difference could arise from the difference in trade accounting rules. Our results show a reasonable first order approximation to the order matching algorithm that are used by the Nasdaq exchange.

In Rama Cont paper, using the parameters estimated by the authors, we could derive, as a first order approximation, the ratio of the cancellation orders to the total number of orders. The ratio is calculated to be about 33%. Using the Monte Carlo simulation, which is discussed later, we estimated the ratio to be about 40%. Using real data, for Intel on March 09, 2009, our order matching engine had recorded about 20% cancellation ratio. These numbers are roughly on par with each other, which provides some sanity check. We also observe that the estimated parameters in Rama Cont's paper overestimated the ratio of cancellation orders, which does not reflect the feature of our data.

Remark: We've also done extensive research on the TAQ data and compared with our Nasdaq data. While TAQ data are popularly used in the market microstructure field, they are inferior to our data. The most important drawbacks are (1) TAQ data are only precise up to 1 second. (2) TAQ data are Level 1 data with only best bid and best ask. (3) TAQ data doesn't have trade directions and researchers commonly have to resort to Lee & Ready algorithm which provides only about 80% accuracy in trade classification. Therefore we've decided to focus on the Nasdaq data.

3 Stylized Facts of the Data

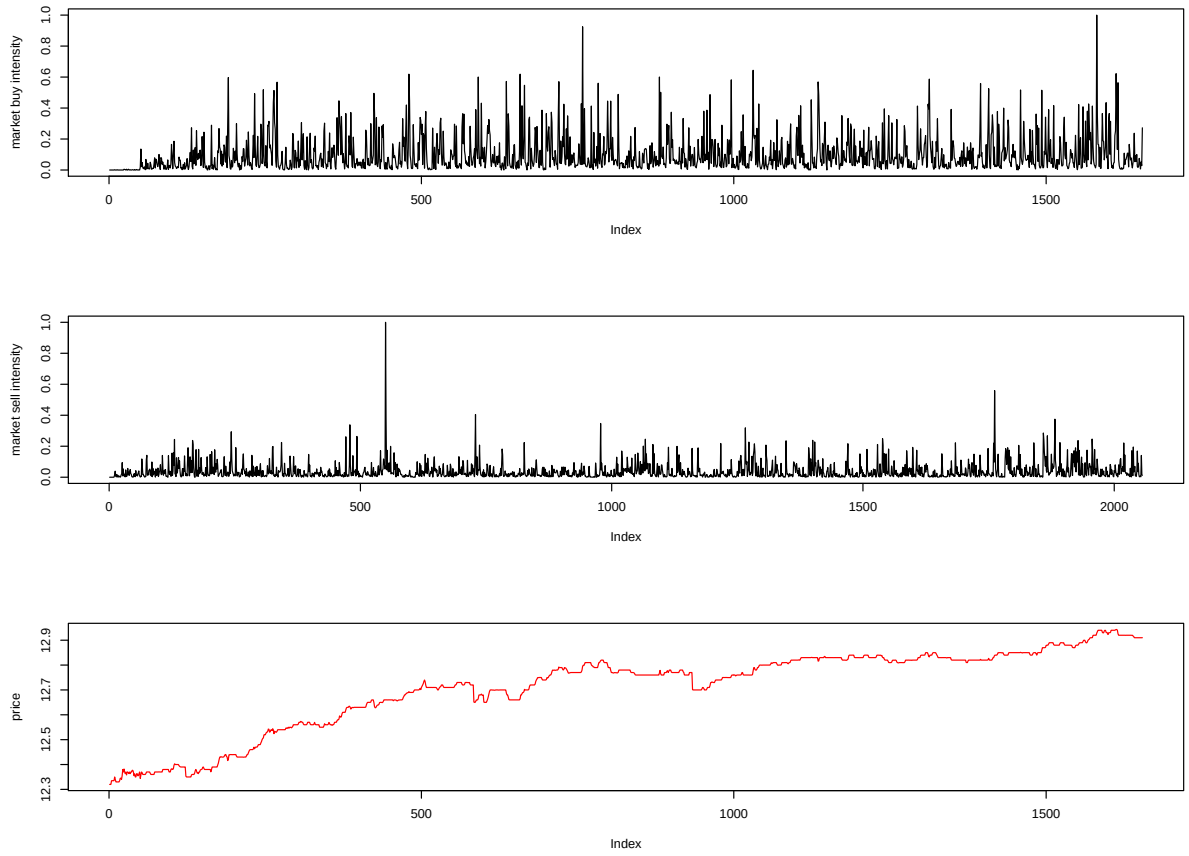
We adopt a data-driven approach and want to design a model that captures the stylized facts of the data. In the sequel we present a few observations about our data, which are the outcome from the the order matching engine discussed above.

3.1 Time Series Properties

The order arrival rates are supposed to influence the price movement. The following plot shows the count of arrival orders per second for market buy(sell) along with the price move-

ment. The clustering effect of order arrival is apparent. The plot was generated from transactions of Intel on March 09, 2009.

Figure 1: Clustering Effect



The histogram of count of arrival order per second for market buy(sell) shows that the distributions for buy and sell are different and the sell distribution has shaper tail.

In Rama Cont'paper, they assume that the order arrival to be homogeneous Poisson process and the duration is modeled as exponential random variable. We check whether this assumption is true. If random variable X is exponentially distribution with parameter α,

$$P(X > s) = e^{-\alpha s}, \log(P(X > s)) = -\alpha s$$

which implies that log of the empirical probability $\log(\frac{\text{number of duration} > s}{\text{number of duration}})$ is linearly proportional to s. However the plot doesn't look like linear at all. Hence the poisson process model is not adequate.

We then study the dynamics of log duration of market buy order. The autocorrelation function of the log duration of market buy order exhibits exponential decay.

Figure 2: Asymmetry of Buy and Sell for Intel on March 09, 2009

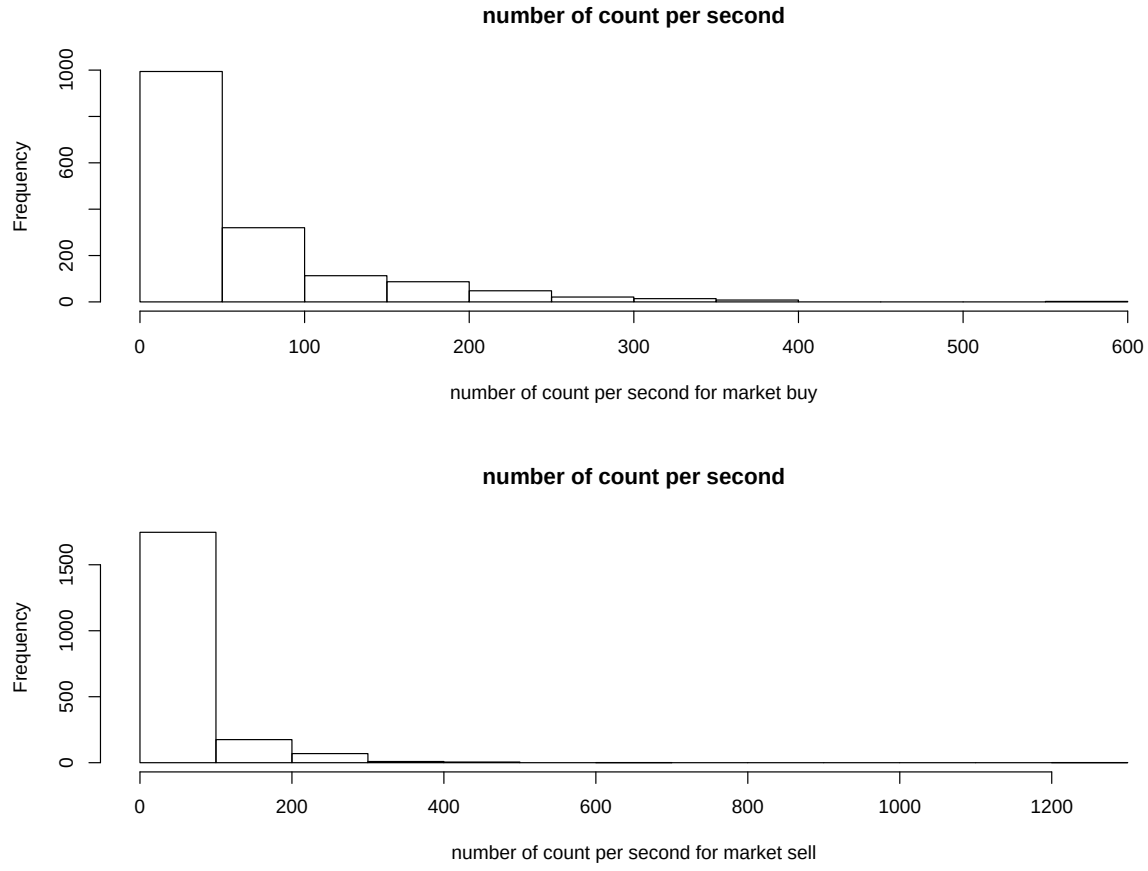


Figure 3: Is Poisson Model Adequate?

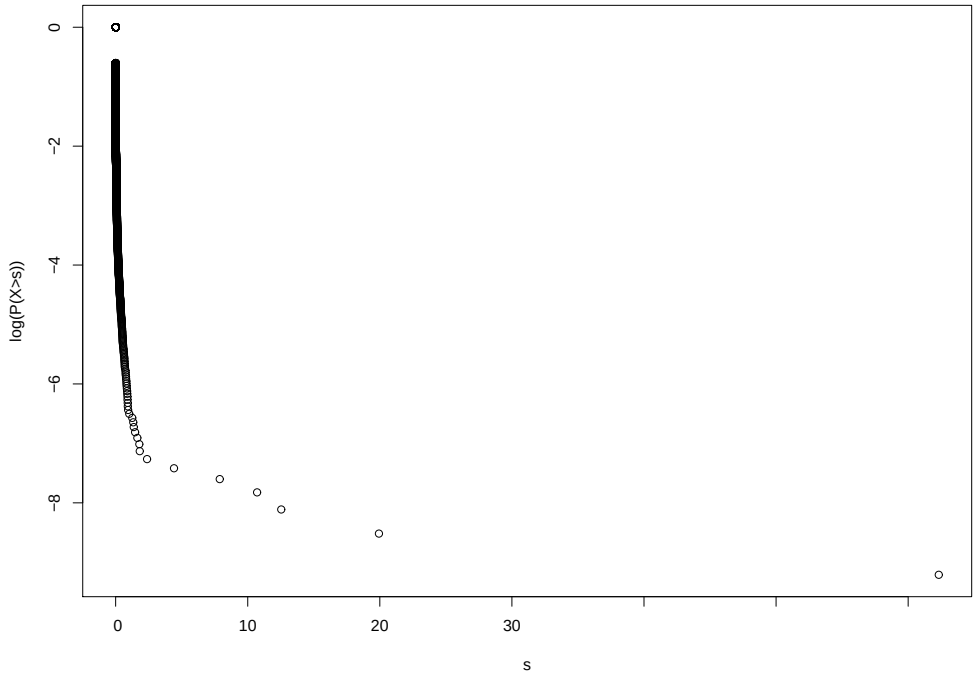


Figure 4: Autocorrelation of the Log Duration

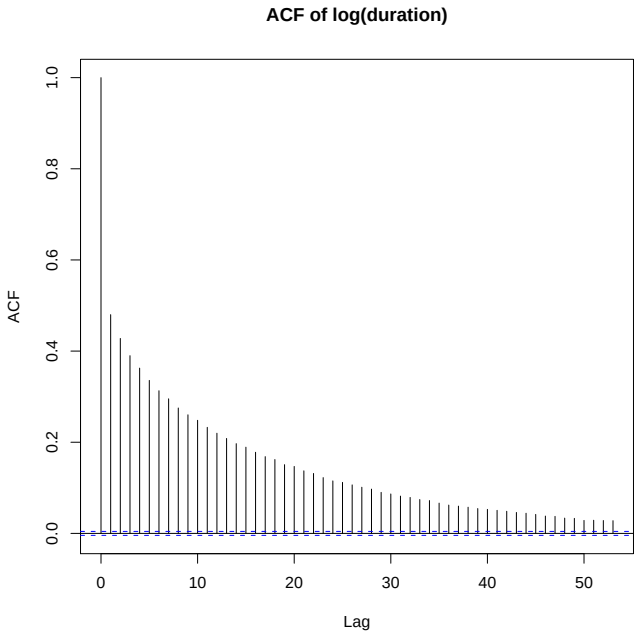


Table 2: The Selected Model using AIC Criterion

Parameter	Estimate	Std. Error	t value	$Pr(>\|t\|)$
(Intercept)	474.09734	323.29557	1.466	0.1432
lastmbdurlog	0.25836	0.04299	6.010	3.59e-09 ***
ltp	119.53096	51.95104	2.301	0.0218 *
vwapb	-158.46802	77.53183	-2.044	0.0415 *

The autocorrelation function of the differenced log duration indicate that AR(1) model might be appropriate here.

To see whether other covariates have effects on the log duration, we regress the log duration on the last log duration,spread, current ask(bid) price, ask (bid) volume, last trade price, volume weighted average bid (ask) price and etc. 500 data points are used which cover time period [09:31:37 AM - 09:33:35 AM 09/03/2009]. By using AIC criterion, following model is selected. The relationship evolves over time.

3.2 Empirical Probabilities of Mid-Price Movement

Since later on, we are going to present our trading strategy which is based on the mid-price movement. We investigated the empirical distribution of the mid-price uptick movement, conditional on the current order book status.

In the table, the b's denote the number of quotes outstanding at the best bid level.

Figure 5: Autocorrelation of the Differenced Log Duration

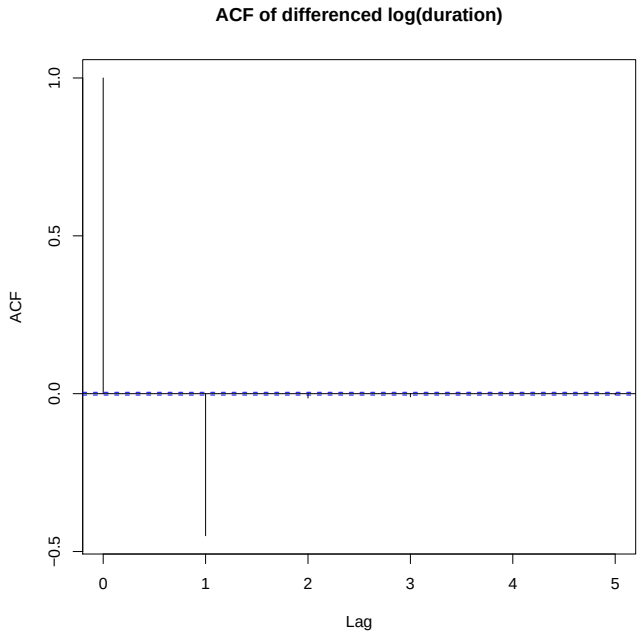


Table 3: Empirical Distribution of Mid-Price Uptick Movement

	a_1	a_2	a_3	a_4	a_5
b_1	0.5	0.5737	0.6607	0.8862	0.4615
b_2	0.4321	0.5248	0.372	0.9533	0.8767
b_3	0.391	0.3804	0.4646	0.9415	0.7857
b_4	0.8031	0.3488	0.3587	0.8438	0.6857
b_5	0.2522	0.2805	0.3662	0.6829	0.9297

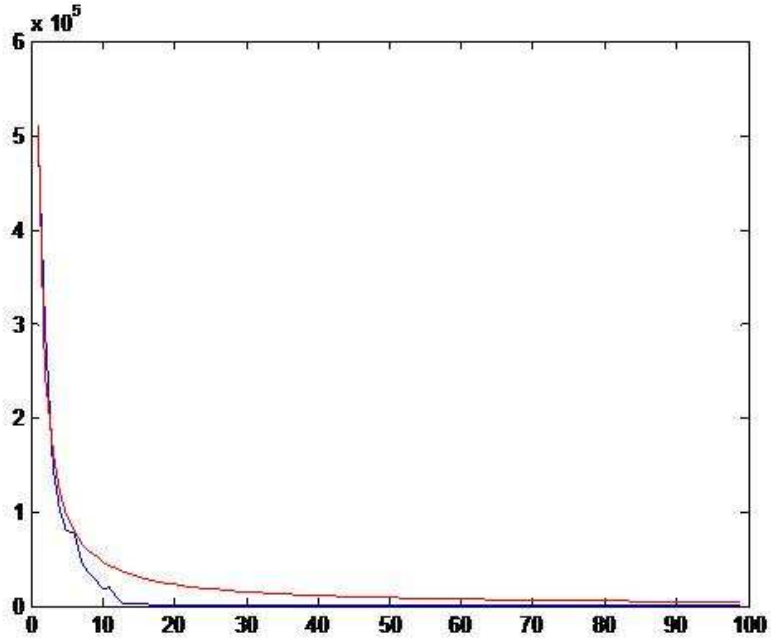
For example, b_1 represents the status where there is 1 quote outstanding at the best bid, b_2 represents the status where there are 2 quotes outstanding at the best bid, so on and so forth. Similarly the a's denote the number of quotes outstanding at the best ask level. In this table, the (b_3, a_1) element represents the probability of uptick mid-price movement, given there is a mid-price movement, conditional on the current order book status, where there are 3 quotes outstanding at the best bid and 1 quote outstanding at the best ask. The above table also conditions on the current bid-ask spread being 1. The reason for this setup will become clear later in the discussion of the trading strategy.

The table was obtained by going through 10 days of Intel data, from March 09, 2009 to March 20, 2009, and computing such probabilities empirically. We observed that the number of samples are not sufficient for a good convergence. In each single day, there are only very few samples of the number of best bids and best asks to be exactly in between $1, 2, 3, 4, 5$. In Rama Cont's paper, they've computed the empirical distribution from about 125 trading days.

3.3 Stylized Fact about Order Book Dynamics

According to various literature studies, the arrival rate of the orders satisfies the Power Law, where the rate of the order arrivals is a power function of the distance between the order price and the opposite best quote price level. The distance here is measured in ticks.

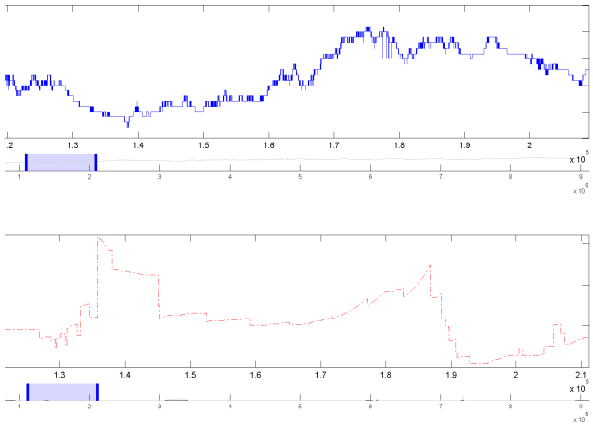
Figure 6: Power Law of Order Book



The red curve shows the fitted the Power Law model. The blue curve is the real empirical result from the Intel data on March 09, 2009. It's seen that the Power Law does not indeed

provide a good fit, because it leads to very fat tail. In fact, we have been able to fit the model better by the exponential functions, somewhat an Exponential Law.

Figure 7: $\frac{BuyingPressure}{SellingPressure}$ vs. Price Movement

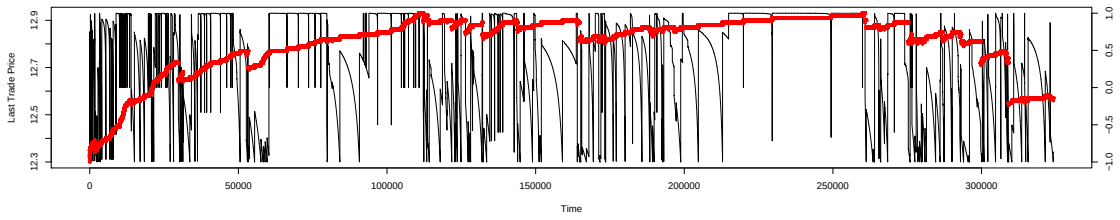


In Fig. 7, the upper panel shows the price movement, whereas the lower panel shows the indictor defined by $\frac{BuyingPressure}{SellingPressure}$, where the Buying Pressure and Selling Pressure are defined by total outstanding quotes at all levels of bids and all levels of asks, respectively.

It could be seen from the figure that in the market places there are two types of traders interacting with each other: trend following traders vs. contrarian traders. When the buying pressure and the selling pressure are balanced, or only slightly skewed, it's often a good time for trend followers. When the buying pressure is extremely high compared to the selling pressure, the contrarian traders will grow stronger and stronger, and once $\frac{BuyingPressure}{SellingPressure}$ exceeds the critical point, the price will dramatically shift its direction and the above cycle will go on and on again.

The above observations form the foundation of intraday trend-following and mean-reverting based trading.

Figure 8: Selling Power vs. Price Movement



In Fig. 8, the price and the sell power are plotted together. Here the sell power is

defined to be $\frac{SellingPressure-BuyingPressure}{SellingPressure+BuyingPressure}$. The slow decay in the midrange of the plot might be caused by the large intuitional investors, executing their large orders judiciously. We are still investigating the causes of the jumps.

4 The Birth-Death Chain Model for Order Book Evolution

Here we briefly review the model introduced by Conte et al for predicting the movement of best bid and ask prices. We shall call the collection of outstanding orders at each price level the stack at that level. By convention, the size of the stack is a nonnegative number, hence a limit bid order is *not* considered a negative order. The terminology will be extended naturally in future work to the notion of a nondeterministic priority queue, which accounts for the variability of order size in terms of number of shares. For each stack, the evolution of its size is assumed to follow a birth-death process, namely a discrete state space continuous time homogeneous markov chain which behaves locally as following: for $X_0 = x_0$, and $t \leq \tau := \inf\{s > 0 : X_s \neq X_0\}$,

$$X_t \sim x_0 + D(x_0\theta(p) + \mu 1_{p=p_A||p=p_B})_t + B(\lambda(p))_t$$

where D is a death process (i.e., Poisson process with negative increments and rate $x_0\theta(p)$) and B is a birth process with rate $\lambda(p)$, which are both functions of the price level. The reason the death rate is proportional to the stack size is simple: each outsanding order has equal likelihood to be cancelled by its issuer, and for modeling purpose we might as well assume they are independent. p_A, p_B stands for the best ask and bid prices respectively.

In [1], the arrival rate of limit buy orders and limit sell orders are not distinguished. Thus they did not introduce two different rate parameter functions λ^A and λ_B. This has been found inconsistent with empirical estimation, hence we propose in our implementation two different rate functions. Also it was observed in [1] that the arrival rate function $\lambda(p)$ satisfies roughly a power law, i.e.,

$$\lambda(p) = c(p - p_O)^{-r}$$

where p_O denotes the best opposite price (so if p is an ask price level, then p_O denotes the best bid order and so on), in reality we noticed that the rates have rather thin tail, hence better resembles an exponential trend. Therefore in our factor model for the arrival, cancellation, and market order rates, we use the following generic form to calibrate the statistical relationship between the various rates and the relevant covariates:

$$\lambda(p) = \exp(\sum_{i=1}^{N} \beta_i Y_i)$$

where the Y_i are covariates to be chosen later, one of which should be p itself. Typically these can be macrolocal variables such as VIX index, or micro information such as the shape of the orderbook at the current time second or with time lag, and β_i are the dynamically calibrated parameters to be determined by MLE.

The effectiveness of this strategy is determined by the order arrival rates. Suppose in the above scenario the sell order rates are much larger than the buy rates, then a seemingly favorable order book may actually be the opposite. This motivates modeling the order rates as a function of time-varying covariates. Let $\lambda_t(p) = \exp \beta^T \mathbf{x}_t$, where $\beta \in \mathbb{R}^p, \mathbf{x}_t \in \mathbb{R}^{T \times p}$. Under this model we assume that the inter-arrival times orders are exponentially distributed with rate $\lambda(p)$. One would model each order rate separately, both by type (limit/market, buy/sell) and by distance to the current bid/ask.

To fit this model let $y_i \in \mathbb{R}$ be the order inter-arrival times, then the log-likelihood is

$$\log L = \sum_{i=1}^{n} \beta^T \mathbf{x}_i - y_i e^{\beta^T \mathbf{x}_i}$$

and the MLE estimate for β is the solution to

$$\frac{\partial \log L}{\partial \beta_k} = \sum_{i=1}^{n} x_{ik}(1 - y_i e^{\beta^T \mathbf{x}_i}) = 0.$$

To find β we use Newton's method since

$$\frac{\partial^2 \log L}{\partial \beta_j \partial \beta_k} = -\sum_{i=1}^{n} x_{ij} x_{ik} \underbrace{y_i e^{\beta^T \mathbf{x}_i}}_{\geq 0}$$

hence the Jacobian matrix, J, is positive semidefinite. Therefore β is given by the iteration

$$\beta_{n+1} = \beta_n + J^{-1}(\beta_n) \left. \frac{\partial \log L}{\partial \beta} \right|_{\beta_n}$$

until convergence.

Typically the covariates used in the factor model can be macrolocal variables such as VIX index, or micro information such as the shape of the order book at the current time with time lag, e.g. lagged arrival times, $y_{i-1}, y_{i-2}, \ldots$ etc., as well as the momentum indicators such as the change in price level over the last few orders and/or seconds. In addition one could consider functions of the order book itself such as imbalance between the shares at the bid and ask.

After a universe of candidate predictors is selected one must do feature selection. Since this is a simple model (low variance, potentially high bias) we need to re-fit parameters very frequently. A simple and relatively fast feature selection algorithm we considered was stepwise forward selection via AIC. This requires us to add the variable that improves the likelihood the most, stopping when the AIC increases where $AIC = -2 \log L + 2k$ and k are the number of fitted parameters. The stepwise approach has the same flavor as model selection via ℓ_p ball method, which is detailed in §4.1.

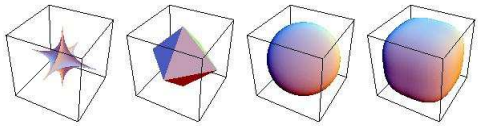


Figure 9: Illustration of ℓ_p balls when p increases. The unit ℓ_p balls are more spiky when p is smaller which is good for sparsity

In practice we found that the β estimates are highly variable over even very short time-intervals so stability is a major issue. This makes a seemingly low variance model perform very poorly. One correction is to use a moving average of β estimates as we re-fit the model for prediction purposes. Given our model the hope is to compute more realistic order arrival rates and consequently more accurate probabilities for an increase in the midprice.

4.1 Feature selection of time-varying rate model

As mentioned in §**??** we could have a large pool of covariates to be included in the model estimation. This inevitably increases the computational cost and more important may cause the instability of parameter estimation and the over-fitting to the data. As a simple approach, AIC can be used in the model to choose the most relevant features automatically step by step. Here we introduce two methods which are often taken as the robust approaches: principal component analysis (PCA) and ℓ_p based regularization.

PCA is a subspace method which projects data into a lower dimensional space. It is purely data based analysis which is independent of the model to be built for the data. Using PCA, the important covariates are chosen as linear combination of available features that can explain reasonable large portion of variation of underlying data. To explain a large portion of variance may result in taking most of the covariates feeding into the problem. Computing subspace projection of data can be time consuming. Also we may need to face the problem that in frequent fitting of parameters, the number of observations in each data chunk could be less than number of the available features. To reduce the two shortcomings, usually it is desirable to apply sparse transform to the data using for example Fourier transform or wavelet transform. Then PCA is done on the transformed data which usually should be very sparse, i.e. only very few entries are nonzero for each feature. By doing so, computational cost can be extremely reduced. Then the principal components are obtained by applying inverse sparse transform to the components obtained over sparse data.

Another approach is considering the regularization together with parameter estimation. The popular regularization terms are ℓ_p balls, especially for $p \leq 1$ in order to conserve sparsity of the estimated parameter. See Figure 9 for an intuitive explaination. The sparsity of the estimated parameter guarantees that the only limited number of features is to be used in the model while at the same time a large portion of variance could be explained. For our model, with regularization, we have

$$\widehat{\beta} = \arg\min\{-L(\beta; \{\mathbf{x_i}\}) + \lambda \|\beta\|_{\ell_\mathbf{p}}\}.$$

When $p = 1$, we have the iterative solver for the ℓ_1 regularized problem in terms of soft-

thresholding as follows:

$$\sum_{i=1}^{n} \beta^T \mathbf{x_{ik}}(\mathbf{1} - \mathbf{y_i} \mathrm{e}^{\beta^{\mathbf{T}} \mathbf{x_i}}) - \lambda \operatorname{sgn}(\beta_{\mathbf{k}}) = \mathbf{0}.$$

Path parameterized by threshold operator λ can be plotted. It could be useful to gain some intuition when the covariates become related to the problem. Compared to PCA, regularized methods are combined with parameter estimation and also have the stepwise flavor as AIC.

4.2 Simulating the Probability of Midprice Increase

When the order rates are time-varying probabilities must be computed via Monte Carlo. A simple algorithm is as follows. There are six types of orders; buy/sell and market/limit/cancel. For each type of order there are multiple rates depending on the distance to the bid/ask, i.e. if the bid is at the tenth highest tick level then there are ten limit buy orders.

Let $\lambda_i, i \in \mathcal{I}$ be the collection of all order rates and $\mathbf{x}_t = (x_1, \ldots, x_n)$ be the current state of the order book, as specified in [1]. Then there are a fixed and finite number of possible states $\mathbf{x}_{t+1}$ can take on. The next state of the order book is completely determined by which order arrives first. It is known that if $X_i \sim exp(\lambda_i)$ then

$$\mathbf{P}(\min\{X_i : i \in \mathcal{I}\} = X_k) = \frac{\lambda_k}{\sum_{i \in \mathcal{I}} \lambda_i}.$$

Therefore to determine the next state of the order book we just sample $u \sim \mathcal{U}(0,1)$ then partition the interval $(0,1)$ according to the above probabilities to determine which order arrived first. After the next state of the order book is computed we recompute the λ_i's since they depend on the order book, i.e. x_t is an inhomogeneous Markov chain, and repeat to generate an entire sample path.

Let A be the set of ω where the midprice increases, to compute its probability we simulate sample paths until there is a change in the midprice and compute $I_A(\omega)$ then estimate the probability as

$$\frac{1}{N} \sum_{i=1}^{N} I_A(\omega_i).$$

5 Trading Strategy

Following the framework of [1] our strategy is to buy at the ask and wait for an increase in the bid beyond our original purchase price. The smaller the spread the more favorable. Consider when there are x orders at the bid and 1 at both the ask and one tick higher, with a spread of 1. This scenario is the most favorable for our strategy especially when x is large. Once we submit a buy order the spread widens to 2 and the order book dynamics makes it likely there will be an increase in the midprice.

The true predictive power of the birth-death model introduced in [1] comes in when the spread between the best bid and best ask prices are rather small, typically of size 1. When this happens there is a significant probability that the best bid price will move beyond the best ask price in some forseeable future instant of time, provided that there is intuitively speaking a predominant buying pressure in the market. The latter buying pressure condition can be typically stated as following: the best bid stack has size greater than or equal to 3, whereas the best ask stack has size 1 and the second best ask stack has size 0. By second best ask we mean one tick above the best ask price, not the second nonempty ask stack.

It is important to note that since our trading strategy is based on market orders, i.e., we do not assume the role of a market maker who is primarily interested in placing limit orders, the only way to generate positive profit is by "'crossing the spread"'. This latter term means the bid price we sell at must be higher than the ask price we bought with. So we would naturally prefer smaller spread size. Indeed Cont's paper also calculates the probability of making the spread, namely when one places both a bid and an ask limit order on the market and calculate the probability that both get executed before the mid price moves. We cannot however test such strategy based purely on orderbook information. So instead we chose to stick with the market order based strategy.

The basic trading cycle is defined as follows. When the spread is 1 tick, and there are more than 3 orders at the best bid, exactly 1 order at the best ask, and less than 1 order at the second best ask, we place a market buy order to clear that 1 order at best ask so that we are in the situation of spread = 2, greater than 3 best bids, and less than 1 best ask. According to the calculation with most estimated parameters for model (i.e., the arrivial rates, cancellation rate, etc), the probability of the mid price moving up is significantly greater than 0.5. Hence there is a good chance that the best bid price will move up, and hopefully by more than 1 tick. An exit strategy is defined by There are two potential problems with this strategy. First of all it is unclear what the actual probability for the bid to move up is. Indeed the mid price movement could also be caused by someone matching the second best ask, which has a reasonably high probability since there is fewer than 1 orders at that level immediately after our market order gets fulfilled. Secondly the strategy delineated above basically reduces to a form of technical trading rule, since it can also be viewed as a consequence of overwhelming buy pressure, which is independent of the theoretical underpinning. The authors in [1] did not test their strategy on real price series. Instead they simulated the price movement based on the birth-death chain model and obtained a small marginal profit in the end. In the sections below we have attempted to test our strategy on real market data, and the results were not as favorable. The testing procedure against real market data also poses the problem of self-interference, meaning that an order we placed to the market could disrupt the subsequent orderbook structure.

5.1 A Preliminary Study on the Conditional Probability of Mid-Price Increase

Below we have used one day worth of best bid and ask information extract from the orderbook data of Intel Corps. In an attempt to statistically justify the 3 vs. 1-1 trading rule, we calculated the total number of occurrence of x orders at the best ask price and $3x$ orders

at the best bid price and the number of uptick movement conditional on such event. The results are tabulated below and show that indeed the probability of uptick is signficiantly above the average, which would be 1/2.

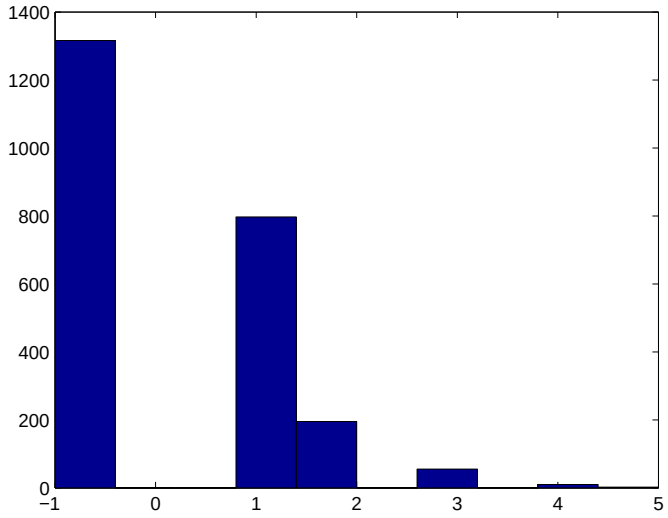
```
 [,1] [,2]
 [1,]   33   29
 [2,]   10    4
 [3,]    6    4
 [4,]    5    2
 [5,]    3    2
 [6,]    4    2
 [7,]    0    0
 [8,]    1    0
 [9,]    1    0
[10,]    1    1
```

5.2 Simulated Results

In [1], the authors didn't provide P&L results on real data. Instead, they simulated the order arrivals based on their calibrated model, and executed their trading strategies using the model generated data.

We replicated their results and used the simulated P&L as a first sanity check of our implementation, as shown in Fig. 10.

Figure 10: Monte Carlo Simulated Order Arrivals and P&L of Our Trading Strategy



From the simulated P&L, it shows that had the data been generated by the calibrated model, the statistical arbitrage trading strategy works. Although there are a large number of

trades in which we lose by one tick, in expectation, in the long run, by law of large numbers, we have a positive return. The cumulative wealth has a positive drift upwards.

5.3 Results on Real Data – Static Testing

As with all kinds of trading strategy back-testing, there is one issue warrants special care. When we decide to enter a trade as well as exit a trade, our trade should be as real as possible. In the current setting, our trading strategy requires that when entry signal arises, we issue a market buy to knock off the outstanding best ask quote, and raise the mid-price immediately to at least one tick higher. The whole dynamics immediately changes after we submit the market order and we have to take such price impact into account, otherwise our P&L calculation is effectively invalid, especially for trading strategies that are very sensitive to the tick movement.

One way to circumvent to problem is to interactively update the orderbook by taking our orders into account and match the bid and ask orders based on the updated information. However dynamically updating the orderbook poses additional computational challenge so we postpone the discussion to the next section.

Here we proceed as though our trades are such that the price impact are minimal and the orderbook are static in the sense it does not reflect our submitted quotes at all. This is also called "passive" or "non-intrusive" back-testing, which is okay when the price impact is reasonably small. To proceed, we made simplifying assumptions that each order arrival is of unit size 1, and therefore there is no cascading effect when we match our orders with the orders that are already in the orderbook. The price level will at most shift by 1 tick.

The backtest including the MLE and Monte Carlo simulation was done in Matlab where we used, statically, the orderbook information extracted from the raw Intel data on March 09, 2009, using the previously mentioned order matching engine as described in Section 1. Our covariates used are firstly [LastTradePrice, VolumeWeightedAskPrice, AskVolume, VolumeWeightedBidPrice, BidVolume] as well as the autoregressive lagged order arrival intervals. We estimated the rates separately for [MarketBuy, MarketSell, LimitBuy, LimitSell, CancelBuy, CancelSell]. The MLE estimation is slow but reasonably okay under Matlab. The P&L results are not very favorable and we've explored various covariates. It seemed that the covariates are not the most essential elements that affect the performance of the model and the trading strategy, so we shrank down to use only [LastTradePriceChange, VolumeWeightedAskBidPriceDiff].

As always, there is a threshold for deciding about when to enter a trade. The threshold is the only part that's not so black-box. The best way to select such threshold is to do historical back-test. But that's under the assumption of data stationarity and ergodicity, which is often times unrealistic for data with regime switching, especially in today's highly volatile trading environment.

Fortunately, in this project, the nature of statistical arbitrage renders the threshold level less sensitive to data. We could pick some value that's reasonably above 0.6. Again in our testing, the threshold is not the key factor, i.e. the trading strategy lost money consistently, but even if we raise the threshold, it's still losing money, as evident from the P&L plots. Of course if we raise the bar to exceedingly high, we end up being so conservative that we don't have any trades during the whole trading session.

Figure 11: P&L of Our Trading Strategy on Intel Data March 09, 2009

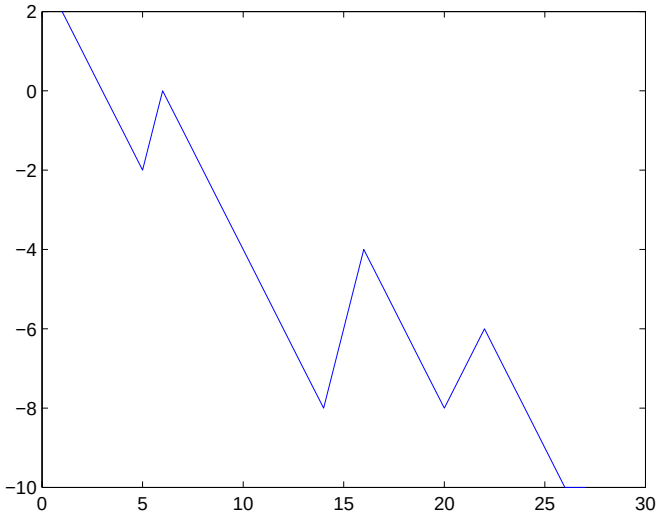
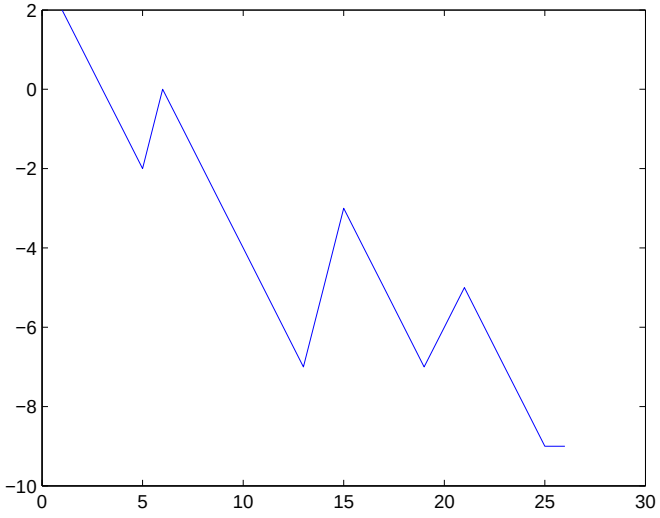


Figure 12: P&L of Our Trading Strategy on Intel Data March 09, 2009, with Raised Threshold



5.4 Results on Real Data – Dynamic Testing

As we described in Section 1, with our order matching engine, in theory, we could dynamically interleave our test trades into the real-data streaming order arrival messages and therefore allow our test trades to dynamically interact with the historical order flow and thus price impact could be studied. In theory, we could use this engine to study the property of VWAP and the best price that one could obtain when executing a large block of trades.

However, the big problem is the computational challenge. And the key bottleneck is keeping track of and matching of the cancellation orders. We have to keep track of the market makers who submitted the exact same order before and search for that order each time when a cancellation order arrives. Even though searching and table-lookup is already super efficient in light of the development of modern STL and C++ language. It's still extremely slow when we are processing tens of millions order arrivals, especially when this order matching engine is the inner loop of multiple outer back-test optimization loops. The computational time is in the order of exponential growth.

Therefore, we decided to make simplify assumptions for the cancellation orders in order to move on and focus solely on the trading strategies. We assumed that the cancellation orders are processed at the national and aggregate level. The cancellation orders are processed in time order, relaxing the requirement of tracking the same market maker. This assumption is okay because we've already made the assumption that the order arrivals are of unit size, therefore the impact of tracking the same market-maker and cancelling only that particular order is minimized. Furthermore, for QQQQ data, we've found that the number of cancellation orders is extremely small, which is a bit unrealistic, given the liquidity of QQQQ. To further improve the computational efficiency, we assumed that those extremely small amount of cancellation orders are neglible.

The plain dynamic order interleaving for the simple model and trading strategy still was not promising. However, our study of the combination of technical rules and statistical rules in the sequel is based on the dynamic interleaving, due to its superior speed.

5.5 Computational Issues

The key bottleneck is the MLE and the Monte Carlo simulation part. We need the MLE to estimate the parameters and calibrate the model. We need the MC to generate the probabilities for making the bets. We have explored parallel and multithreading computing for the MC part and had dramatically improved the speed.

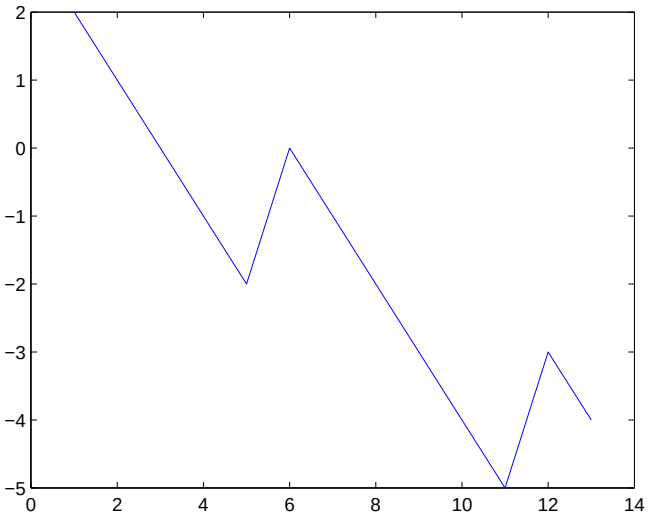
5.6 Improving the Trading Strategy by Combining Technical Rules and Statistical Rules

One observation is that the estimation of instantaneous rates is important. To test this, we used the inhomogeneous Poisson model and estimated the instantaneous rates at every order arrival. The result is shown below:

The previous results show that the 3 vs. 1 statistical trading rule performed poor on real data. Why?

Let's follow the intuition and the big picture at the micro-structure level.

Figure 13: P&L of Our Trading Strategy on Intel Data March 09, 2009, with Instantaneous Rates



The condition is that we enter into a trade when there are more than 3 orders outstanding at the best bid and 1 order outstanding at the best ask and less than 1 order at the (best ask +1) price level.

This simple rule is somewhat overly simplistic, because when the 3 vs. 1 condition is satisfied, there could be two scenarios happening - either the price is trending down, or the price is trending up and there could be jumps in the trends which the paper didn't capture.

Therefore, an extension of the model is to condition on the sequential trend of the orderbook dynamics, instead of a static snapshot at a point of time. We've explored along this direction, the result is better but still not super exciting.

We could think intuitively: what happens when the number of orders outstanding at best ask is 1? Two opposite scenarios can happen: either a price downward shift, or a price upward shift. After a downturn trend is established, the "mood" will dramatically change and reinforce itself so all the rates tend to change rapidly and reinforce the downwarding trend. The clustering effect.

That's because (1) we expect the price being trending up to be in our favor; however, when 3vs1 is satisfied, the price could already be trending down, and we will be swamped in a massive drawdown; (2) when the overall price mood is already trending down, the rates will be nonstationary, much like a regime shifting or volcano eruption. Overall, conditioning on a static snapshot of orderbook is not a good idea; need to condition on the sequential dynamics or the mood of the orderbook.

One way to improve the trading strategy is to gauge more information out of the orderbook dynamics. Recall that in Section 3, we have discussed the stylized facts of the orderbook dynamics. And we have seen that the buying pressure and selling pressure are useful indicators of the price movements. We have defined one indictor to be $\frac{BidVolume}{AskVolume}$.

Our technical rule is simple, when the indicator is in our favor, we issue buy and sell signal. There is a threshold, and the threshold needs to be tweaked based on the fundamentals and broad markets. For example, March 09, 2009 was not as profitable as March 10, 2009. March 09 is a range-bounding because analysts said Intel was going to meet the expectation however the broad markets were full of uncertainty. March 10 was more profitable for trend followers because Citi and the Wall Street surged.

Therefore, our trading strategy works best when we incorporate market sentiment and fundamentals, to decide the "mood" of the stock of that day. The grand view about the markets is captured in the threshold of the technical rules. We could also view this from another angle, at each day, the trader who is experienced with the markets need to make adjustment about the threshold based on the broad markets and the individual stock. If there is positive or negative news, the trend follower will jump into the game, so will be the threshold-adjusted technical and statistical trading algorithms. Viewed yet from another angle, each day, we could select the worst performers or best performers from the previous day, and then the mood of these stocks are more certain than others.

One piece of trade log for Intel on March 10, 2009 is shown below:

```
2009-03-10 04:12:05.423881054, 1236672725.0, ARCX, 12.59, 10, 12.94, 5
2009-03-10 09:32:50.283174992, 1236691970.0, BATS, 12.89, 40, 12.91, 56
PAndL = 1
2009-03-10 09:32:59.188330889, 1236691979.0, BATS, 12.85, 57, 12.87, 56
2009-03-10 09:33:17.595057964, 1236691997.0, ISEC, 12.88, 18, 12.9, 26
PAndL = 2
2009-03-10 09:53:11.480942965, 1236693191.0, CINN, 13.05, 2, 13.07, 2
2009-03-10 09:53:54.011337042, 1236693234.0, ARCX, 13.08, 89, 13.09, 84
PAndL = 3
2009-03-10 09:56:34.127204895, 1236693394.0, ARCX, 13.19, 46, 13.2, 74
2009-03-10 09:59:55.130388975, 1236693595.0, BOSX, 13.1, 2, 13.13, 2
PAndL = 4
2009-03-10 09:59:55.132678986, 1236693595.0, ISEC, 13.12, 50, 13.13, 56
2009-03-10 10:03:02.776428938, 1236693782.0, BOSX, 13.12, 2, 13.15, 2
PAndL = 5
2009-03-10 10:21:15.268723965, 1236694875.0, BATS, 13.37, 110, 13.39, 105
2009-03-10 10:21:15.268723965, 1236694875.0, BATS, 13.37, 110, 13.39, 106
PAndL = -1
2009-03-10 10:21:15.268723965, 1236694875.0, BATS, 13.37, 110, 13.39, 106
2009-03-10 10:26:14.503158092, 1236695174.0, FLOW, 13.3, 21, 13.34, 3
PAndL = 8
2009-03-10 14:18:45.309432983, 1236709125.0, CINN, 13.52, 4, 13.54, 2
2009-03-10 14:25:17.640872955, 1236709517.0, BATS, 13.54, 119, 13.55, 166
PAndL = 9
```

It's interesting that the 03/09/2009 was a up day, with a lot buy pressure than the sell pressure, however Intel closes slightly up, and the market sentiment had been week.

That's what we call the "mood" of the market and the "mood" of the stock.

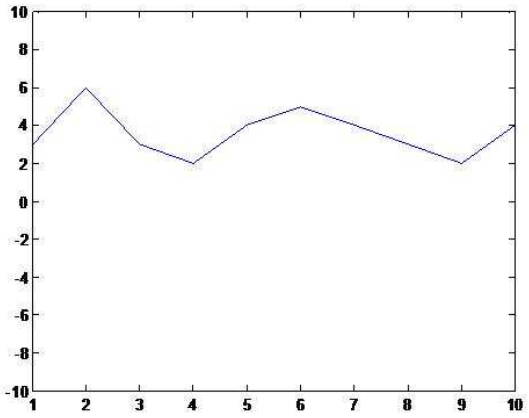
An experienced trader, after testing the water for a while and making a few mistakes, will immediately recognize the "mood" of the broad market and the stock.

Assuming we sense our the "mood" is weak. And we go short more than long. Here is a result that combines the simple technical rule and the statistical trading rule.

Here is another piece of trade log for Intel on March 09, 2009.

```
PAndL = 3
PAndL = 6
PAndL = 3
PAndL = 2
PAndL = 4
PAndL = 5
PAndL = 4
PAndL = 3
PAndL = 2
PAndL = 4
```

Figure 14: P&L of Our Trading Strategy on Intel Data March 09, 2009, with Tech Rules and Statistical Rules Combined



As we can see, the result is quite good. This is essentially an expert system where the experts vote about their opinions. Here the technical agent and the statistical agent both vote. And then we enter into a trade.

6 Future Extension

6.1 Extension of the Birth-Death Chain Model

One variant of the birth-death chain model for orderbook dynamics is to incorporate order size as part of the parameters as well. One could for example assume a geometric distribution

of order size, much as is done in the Rydberg-Shephard model. This will introduce extra parameters to estimate, and would require the number of share information for each order. From there on, there are two assumptions one could make on the trading rules with regard to varying order sizes. The most typical rule is the all-or-nothing order matching mechanism, in which a market order is matched with a limit order only when the sizes of both orders match. Under this assumption one would need to keep track of a dynamic prioritized queue of orders at each of the price level. Since orders with bigger size have higher priority of being matched, but the priority also takes into account the time of arrival of the order, i.e., matching follows a order biased FIFO system, the best strategy is to assign some numerical priority value to each order in the queue. To make sure that the arbitrariness of the priority system we are imposing on the orderbook data does not affect the true outcome of the matching, it's best to make the process nondeterministic. Thus one would be talking about a nondeterministic priority queue model for the orderbook dynamics. One simplifying assumption one could make is to describe the distribution of the order size based on its logarithm, most conveniently under base 10. Then one could roughly divide the order sizes into big groups, hence reducing the amount of information to keep track of. Based on such assumptions, one could limit the size of the queue to about 6 or 7 since there are hardly any order of size exceeding 10^7 shares, and only have to keep track of the number of orders of each broad size category. This is of course under a further assumption that FIFO is completely thrown out of the window, which could actually be what's going in real trading, since it's much easier to keep track of the order sizes than their arrival time, and whatever arrives first would have a higher chance of being matched anyway. There will most likely not be any closed form expression for calculating the probability of uptick, downtick and other relevant statistics under such a realistic model. Instead Monte Carlo will be the most convenient approach, and the magnitude of complexity would not be much greater than the simple spatially homogeneous birth-death chain model simulation.

Under a second assumption of order matching, in which partial order matching is allowed, more flexible theoretical tools can be applied. One important setback of even the simple birth-death chain model is that closed form solutions for various probability calculation is hardly available. The Laplace transform methods introduced in [1] were computationally difficult to handle, in addition to being numerically unstable. Such is a common curse with discrete markov chain model, since the calculation usually boils down to some intricate combinatorial expressions, whose simplification is notoriously difficult or in most cases impossible. Thus one idea would be to approximate the discrete model with a continuous stochastic process model, driven by some underlying Brownian motion process. One such candidate is the simplest mean-reverting model, namely Ornstein-Ulenbeck process. Thus the height of each price stack can be modeled as the positive part of a mean 0 OU process, which the reversion speed calibrated in agreement with the death rate and the upward drift proportional to the birth rate. The 0 hiting time of the original birth-death chain should then be correlated with the corresponding 0 hitting time of the approximating OU process. Unfortunately even for a continuous process as simple as OU, the exact formulae for these hitting time distributions only exist for special cases in which the drift is 0, see [2]. Nonetheless, Monte Carlo methods for varoius continuous processes have been extensively studied and even the drift 0 case can form a useful benchmark formula for more complicated models.

6.2 Price Impact

As we described in Section 1, with our order matching engine, in theory, we could dynamically interleave our test trades into the real-data streaming order arrival messages and therefore allow our test trades to dynamically interact with the historical order flow and thus price impact could be studied. It's going to be very interesting studying the how to optimally execute a large block of trade while minimizing the transaction cost and the price impact.

6.3 Adaptive Trading Strategy

All technical-rule based trading has certain sort of threshold which needs to be fine-tuned. Back-testing plays a role in this process. There is always the danger of over-training, as well as data-snooping. And back-testing based parameter optimization is always carried with the assumption of data being stationary and there will be no regime-switching. However as we have seen from the data this is not the case. That's why people often call this type of trading the computer-assisted human trading. Therefore it's better to devise trading strategies that aren't heavily reliant on threshold and optimized parameters. One approach to this problem is the so called adaptive trading strategy. We have done some very immature preliminary study along this line and here are the very preliminary results (cumulative P&L) for Intel for the 9 trading days from March 09, 2009 to March 19, 2009.

```
-6, 7, 8, 9, 8, 7, 6, 5, 4, 3,
1, 8, 9, 10, 5, 15, 12, 11, 10, 11,
4, 2, 3, 2, 3, 2, 14,
3, 21, 18, 16, 17, 16, 15, 14, 13,
1, -6, -5, -6, -5, -6, -7, -5,
2, 3, 2, 0, 1, -1, -2,
6, 5, 6, 7, 9, 7, 6, 4, 3,
-1, 5, 4, 11, 13, 11, 9,
5, 4, 3, 1,
```

Note: The results are very preliminary and we are aware of the various assumptions we've made along the way.

References

[1] Rama Cont, Sasha Stoikov, Rishi Talreja. A stochastic model for order book dynamics.

[2] Anja Goingen Jaescheke, Mark Yor. Clarification note about hitting times densities for Ornstein-Uhlenbeck processes.

Figure 15: P&L of Our Adaptive Trading Strategy on Intel Data from March 09, 2009 to March 19, 2009