Transaction Cost Transform of Trading Volume Series in a Closed Auction Type Stock Market Model

Lukas Pichl[1], Katsuhiko Hayashi[1], Takuya Yamano[2], Taisei Kaizoji[2]

[1]Division of Natural Sciences
[2]Division of Social Sciences
International Christian University, Osawa 3-10-2, Mitaka, Tokyo, 181-8585 Japan

Abstract

Research work on price making mechanisms in computational intelligence still remains somewhat under-represented, in part because of its short-time scale implications, and in part due to the complexity in modeling bid-offer array books and correlating them to individual market players. Here we report a study on the transaction cost – trading volume correlations in an artificial stock market with closed auction pricing. A signature of particular trading strategies has been found in the characteristics of the exponential decay of total trading volume as a function of the transaction cost.

Keywords: java market model, closed auction, transaction cost, bid-offer arrays, behavioral investors, moving averages

1. Introduction

The majority of the research work on the artificial markets published in the literature does not describe the model used in full detail; neither the readers are provided the possibility to download the source code of the simulation program. While the former is plausible because of the paper length restrictions, the latter meant a serious drawback for the community until recently [1-2]. Given the amount of parameters and particular design features that every market model includes, the results of such simulations could not be reproduced without further knowledge.

To address this drawback, the present study of transaction cost effects in a closed auction market is based on the latest YMYP market model developed in our group. The developer environment is implemented in Java, source code of which is freely available for download [3]. Here YMYP stands for the abbreviation of the names of the first generation of the code developers, namely Yuki, Moriya, Yoshida and Pichl.

It is the purpose of this paper to describe the structure of the YMYP stock market model, and study the effect of transaction costs on trading volumes within this frame of reference. The YMYP model is employed because of its realistic trading system, object based structure, which allows for realistic market simulations and easy implementation of various behavioral types in trading strategies, and convenient graphical user interface (GUI), which allows modification of market parameters during the simulation run.

In particular, here we focus on the importance of transaction costs (a factor, which is often neglected in artificial market models) on the side of the automated order matching system, and the importance of market player individuality (namely the individual speed of learning) on the side of market players.

The work is organized as follows. Section 2 explains the closed auction market model, outlines the simulation environment, main features of the GUI, and components of market player strategies. The transaction cost transform into trading volumes is empirically derived from a series of market simulations in Section 3, which also explains its main features in terms of an analytical model, and provides a short discussion of broader implications to real markets. Concluding remarks in Section 4 close this paper.

2. Java YMYP Market Model

As mentioned above, the YMYP is one of the simulation models of financial markets, noteworthy because of the following points:

- The source code of the model is freely available as a standard [3]
- The model fully implements a realistic automated order matching system
- The graphical user interface allows immediate access to all data produced during the course of simulation

The model implements a closed-auction price making mechanism, such as in the Tokyo stock

exchange, which is an example of order driven market. In order-driven markets, all dealing orders are collected during a short fixed period of time, and then their matching is performed without further intervention from market dealers. The price is determined by balancing the bid-offer order arrays for maximal transaction volume. The allowable price interval determined by the last matched prices for selling and buying transactions. If the order matching is not performed because of too large divergence between buy-order prices and sell-order prices, the price will move in one direction by a preset margin to find a way out of the deadlock..

A dividend is distributed each year to the dealer who owns the shares of the particular stock. The amount of pay times per year is kept steady. The revenues follow a stochastic distribution to which the dealers behaviorally respond.

By default, there are three types of stock in this model, which are labeled as IBM, DELL, and TOSHIBA, respectively. There is no relation to the actual stock of the three companies, though, and a different number of stock titles is possible. Since the YMYP model deals with more than one single stock species, it has a potential to be applied to portfolio analysis and optimization. There are 300 market agents by default. Their number can adjusted via GUI as much as the memory limits allow. The number of agent strategies is unlimited; each strategy is specified by a separate java file containing class definitions.

In order to match the macro-level of the automated order matching system with the micro-level behavioral features of market participants, it is necessary to assume a certain minimal mental model, according to which the players act. This is especially important for testing the default version, since the model allows for non-trivial strategies of any type. To that aim, all market data, including agent history files, strategy files are kept in batches in memory, as well as written to disk. Hence the object design of the model allows for implementation of any strategy that involves market history, information sharing, learning etc. While the YMYP simulation environment is kept as least restrictive as possible for this sake, it must also be capable of determining player actions within the order-matching framework. That is why in the YMYP model, every player (market agent) is equipped with his individual view of the correct price to which all strategies are reduced; let us call it IP for short. The buying-selling behavior is then based on the comparison of the IP value and the actual price in the market.

- If the market price is above the individual estimate price, the agent offers (attempt to sell)
- If the market price is under the individual estimate price, the agent bids (attempt to buy)

The process of buying behavior includes a check of the following equation first,

$$P_m^p < P_c \cap M > P_c(kA_b)$$

Here P_m is the previous day's actual stock price, P_c is the agent's IP, M denotes the amount of dealer's money, A is the desired amount of the stock , and k is a parameter smaller than one, in order to account for risk aversion. The process of selling behavior includes a check of the following equation,

$$P_m^p > P_c \cap \text{Stock Amount} > 0.$$

For the supply and demand matching in a closed auction, the system collects both buy-orders and sell-orders at first. Then, the buy-orders are sorted in descending order of the maximal price, and sell-orders are sorted in ascending order of the maximal price. The buy and sell orders are matched as follows.

$$\text{BuyArray} = P_j^b, P_{j-1}^b, ..., P_1^b \qquad (P_1^b \le P_2^b \le ... \le P_j^b)$$

$$\text{SellArray} = P_1^s, P_2^s, ..., P_k^s \qquad (P_1^s \le P_2^s \le ... \le P_k^s).$$

Here P^b denotes the maximal price a buyer can admit, and P^s denotes the minimum price a seller would accept. This price tag includes also the stock code and the mount of stock for the order.

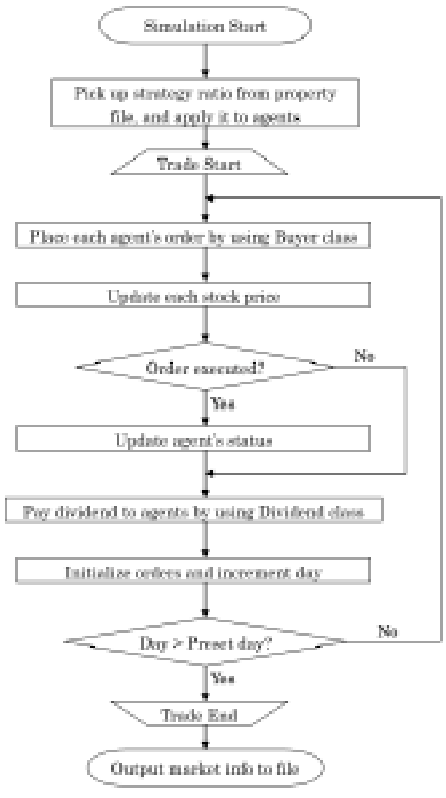

Fig. 1: Flow Chart of the Closed Auction Market simulation in the Java YMYP simulation environment.

By using the above information, order-matching is carried out. The supply and demand matching is started with the lowest bidder (seller requiring the least revenue from the transaction) and the highest bidder (buyer accepting the highest price). The matching is then conducted in this way, sequentially proceeding through the elements of the supply and demand arrays.

The bid and offer prices do not directly determine how the asset of each agent is going to change. This is a more complex process, which includes the previous session's price, the particular structure of the bid-offer book, and the matching described above. After the trading, the resulting changes in each agent's asset and stock holdings are enforced.

Figure 1 shows the flowchart of the entire YMYP framework. The simulation starts with the initialization of all agents. The three basic strategies are distributed to agents by using strategy ratios defined in the property files. At every market round, the agents place orders according to their strategies, then the supply and demand matching is performed, and the new value of stock price obtained from the order book matching. Buying agents decrement their money holdings and increment stock holdings. Selling agents increment their money holdings and decrement stock holdings. If the simulation session meets with a dividend maturity time, a dividend is paid to stock holders by using the Market and Dividend java classes.

3. Results and Discussions

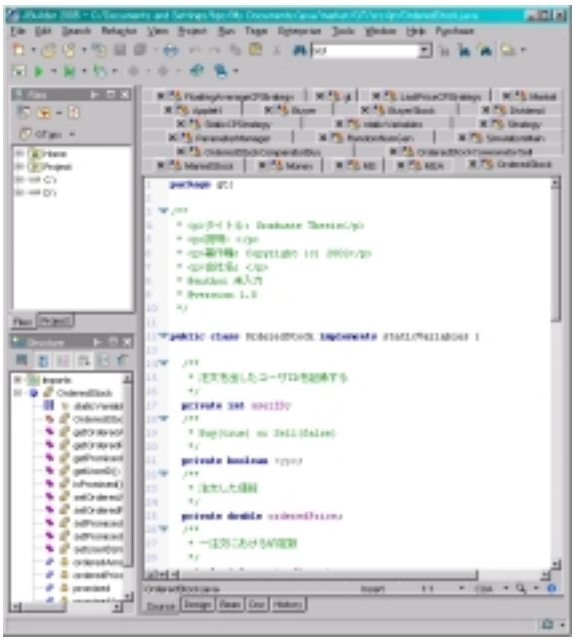

Fig. 2: Java Builder environment for the YMYP model coding. Multiple tabs correspond to new java classes.

We have newly implemented adaptive learning into the three elementary behavioral strategies (fundamentalist, noise-trader, and mental frame) of the YMYP model (cf. Fig. 2 for its Java-Builder environment). The agents in each behavioral group are randomly initialized with an IP value following a normal distribution. Throughout the market, two types of IP beliefs were studied, which are represented by two normal distributions with variances $\sigma_2=2\sigma_1$ and a relative shift of $1.5\sigma_1$ for each stock.

Within this framework, we have performed market-trading simulations over a period of 4,000 sessions. All simulation conditions were frozen except for the transaction cost ϵ, which distorts the market price indicator as $P(1+\alpha)$ and $P(1-\alpha)$. Figures 3 and 4 show the simulation snapshots and the GUI of the system. The overall result of the transaction transform is summarized in Fig 5: the total trading volume in the market for the entire simulation period is shown as a function of the transaction cost.

It is remarkable that this result, which includes three behavioral strategies, closed auction pricing, and a three-component portfolio can be successfully explained in terms of the overlaps of the initial beliefs across market traders, i.e. the IP distribution. The components of supply and demand arrays are proportional to the following intervals,

$$I_j^{\,i}(P,\alpha) = \int_{-\infty}^{P(1+i\alpha)} \frac{1}{\sqrt{2\pi\sigma_j^{\,2}}} e^{-\frac{(x-m_j)^2}{2\sigma_j^{\,2}}}\, dx\,,$$

with i=+/-, and j=1 and 2 (σ_1 or σ_2 variance in the IP distributions). It can be shown that in case of static beliefs and no transaction costs, the curve in Fig. 5 should correspond to $\min(I_1^{-}+I_2^{-},\ I_1^{+}+I_2^{+})$. The two components of the minimum are also shown in Fig. 5. Since the individual agent's IP believes were changing dramatically during the simulation, this is an interesting point for further analysis. The phenomenon has been thus far confirmed in twice as long simulation run and for various values of the parameter σ_1.

4. Conclusion

The Java YMYP model is a new open source platform for agent-based simulations with powerful features in system setup and GUI. Within the closed auction model and stochastic distribution of agent beliefs, it was found that the transaction cost transform strongly correlates with the distribution of agents' IP believes, and can be well approximated analytically in terms of the integral weight of supply and demand functions,

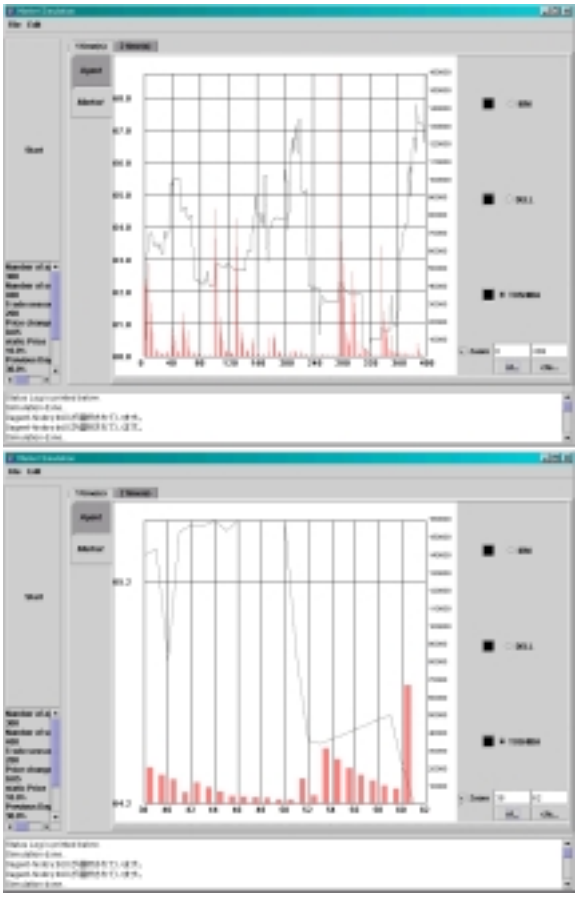

Fig. 3: YMYP market model: macro-level market price (curves) and volume (bars) time series

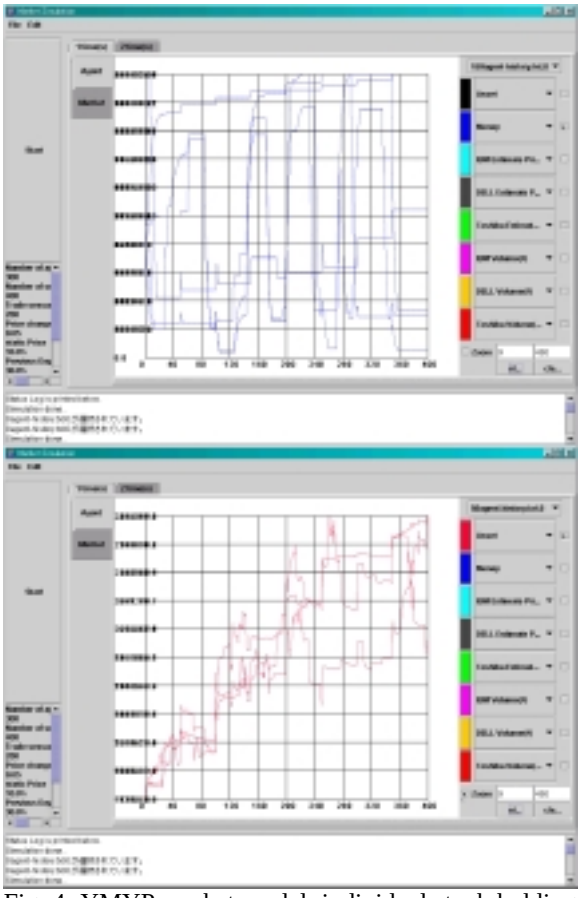

Fig. 4: YMYP market model: individual stock holding (upper panel) and price estimates (lower panel)

and the offset of the two distributions. Thus within the artificial stock market model, the transaction cost transform provides us with an interesting macro-level tool for studying micro-level phenomena.

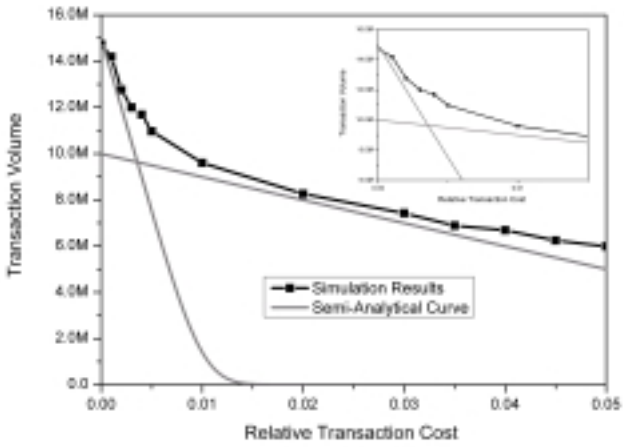

Fig. 5: Transaction Cost Transform of Volume Series

5. References

[1] N. Minar, R. Burkhart, C. Langton, M. Askenazi, "The Swarm Simulation System: A Toolkit for Building Multi-Agent Systems," Santa Fe Institute Working Paper 96-06-42, Santa Fe, NM, 1996. Cf. http://wiki.swarm.org.

[2] J. Hodik, P. Becvar, M. Pechoucek et al., "ExPlanTech and ExtraPlant," International Journal of Computer Systems Science and Engineering **20**, (5) p. 357-367, 2005. Cf, http://agents.felk.cvut.cz/aglobe/.

[3] Source code: http://cpu.icu.ac.jp/YMYP.zip