Zero-Intelligence Economic Trade

Exploring Economic Benefits of Trade with the Auction Extension

By: Mike Tamillow

Abstract

The primary narrative behind all economics is that supply and demand drive market prices. This pattern is explained by increasing prices driving additional suppliers to provide for the market while decreasing prices drive additional sources of demand. In economic parlance, supply side agents are typically called producers, while demand side agents are referred to as consumers. This supply and demand theory models continuous spaces of prices as a function of supply and demand. In many ways this model fails to explain budding markets or wild price swings in times of decreased liquidity.

This paper discusses the creation and implementation of a model built upon Zero-Intelligence Traders (Gode and Sunder, 1993) developed in NetLogo and discusses the creation of an auction extension written in Scala that is used to match all orders at a price by the supply and demand during a clearing event. This paper postulates that the existence and growth of a market can occur independently of outside resources entered into the market, if the supply and demand side components realize a greater than zero-sum outcome to trade. Furthermore, cyclical price movement, market-growth, and a search for equilibrium prices can all be described by a model of trade that is indifferent to the current market prices and focuses only on positive sum outcomes for individual agents. Using multi-agent modeling, we will analyze the fundamental drivers of economic change in a growing market with participants arising from a single producer and consumer. From this model, we will see that much of price movement in trading markets can be explained as a function of supply and demand with consideration to the agent traders.

1. Background

Modeling financial trading markets via mulita-agent simulation is an idea famously implemented in its infancy by the Santa Fe Artificial Stock Market. This project took a bottom up approach to designing financial exchange through agent-based modeling rather than more typical equilibrium approaches. Although this was not the first attempt to model trading behavior, it was perhaps the largest single attempt at the time. The concept took form around Brian Arthur and John Holland, who had a desire to create a co-evolutionary ecology of strategies belonging to traders of different types to let machine learning through genetic algorithms optimize the mix of traders into a stable market. (LaBaron, 2002) This early attempt modeled risk-free bonds and risky stocks to find pricing equilibriums, volatility, and liquidity considerations. It made significant headway in modeling financial markets through heterogeneous, rational agents competing in computational environments.

Auction mechanisms underlying the settling of trading prices and sizes for traders in a market have also been an area of study through multi-agent systems. Simple auctions include English auctions, in which prices are driven up by a single auctioneer who publically announces prices; first-price sealed-bid auctions, in which prices are submitted without public information and the highest bidder wins; and Dutch descending price auctions, in which prices are dropped by a single auctioneer until a buyer takes the current offering price. A more unique auction type is a Vickrey auction, in which the price of the most competitive unfilled bid determines the price of exchange for the top bidder. This mechanism is intended to create honest demand from the participants. (Vidal, 2010) Double auctions fill prices for both buyers and sellers in a market representing supply and demand curves, with specific algorithms to determine who gets filled at what price. A continuous double auction, also called a continuous limit order book, is a time dependent auction which always fills the most recent standing order that matches an incoming order. The auction extension in this paper describes a discrete double auction, a modification of a continuous double auction to find an optimal price given an arbitrary number of buy and sell orders any every price and fill all sell orders below this price, and all buys above, while managing partial fills.

Other notable work has been done in the field of artificial trading markets, contributing to a robust study of financial markets under agent-based modeling. Java Auction Simulator API (JASA[1]) and JCAT[2], an extension on JASA, were developed for Trading Agent Competitions (TAC) which started around the year 2000 to analyze autonomous trading in combinatorial travel markets and supply chain management. The purpose of these competitions was to see how various competitive strategies would integrate into a market environment to determine relative winners in trade. The research also took an in-depth look at strategy interactions, describing how agents with differing strategies would fare against each other in a more minimal game environment. Combinatorial optimization problems in exchange include strategy and asset profiles and the work in TAC provides a strong foundation for evaluation of this category of problem. These small market environments demonstrate a feature of trading domains, that action effects can depend on the actions of other agents, which is a defining property of multi-agent systems. (Wellman, Greenwald, and Stone, 2007)

[1] http://jasa.sourceforge.net/
[2] http://jcat.sourceforge.net/

2. Auction Extension

For the production of a trading simulator for NetLogo there were two options to consider. The first option was that rules of exchange could be produced entirely within the simulation itself, written in NetLogo and satisfied from within the model. The second option was to write an independent extension that would be called from within the model and would do the heavy processing in a lower level language. The second option was more desirable for a number of reasons. Setting a standard for an auction mechanism written in NetLogo, to be reused by others in a variety of models was a significant motivator. The complexities of the algorithm to fill all orders could be isolated in a module that would import and function the same way across different models. The intention is to maintain the auction extension and manage any outstanding issues that may improve the extension over time. This is a general purpose extension written in Scala, which should be able to handle unusual scenarios robustly.

As mentioned, the mechanism that is included in the auction extension is a discrete time double auction. It fills an arbitrary number of agents orders during any clear event, and returns any unfilled assets or currency to the agents' accounts. The primitives that have been included in the auction extension are the commands "setup-market", "buy", "sell", and "clear" and the reporters "last-trade-price" and "last-trade-volume". Keeping the number of primitives to a minimum should be helpful for usability. Setup-market should be called once per market, the market is associated with a turtle, which allows the user to make markets extensible. However, this turtle must be passed to buy, sell, and clear commands to function correctly. Buy and sell commands place orders at a price and a quantity in the market with the respective sides: bid for buy, ask for sell. Prior to orders making it to the market they must check that the price and quantity is greater than 0, and that the current agent placing the order has enough resources. The quantities and price must both be integers. The logic behind this mirrors the logic behind multi-agent modeling. All prices and quantities can be expressed as discrete units. $0.01 can be expressed as one penny, and most quantities in real markets (barring some cryptocurrencies) exist as integers.

The most complex functionality happens during a clear. All orders associated with a market are managed by an instance of the class named “Market” within the auction extension. When a buy or sell happens, it is routed by an object, “Router”, through a HashMap into the appropriate market. An optimal price is found by taking the lowest ask price and highest bid price and, if they cross (bid is greater than or equal to ask), reducing the quantity of both until one order has a quantity of 0 remaining. The fill price would be the one with remaining quantity, if this is the last price to cross. If it is not, then the same process happens recursively with the remainder of the quantity. Infrequent complex pricing scenarios are handled by more detailed mechanisms.

This clearing functionality is another reason that creating the extension in Scala made sense. The activities that are handled by the clear use orders that can be associated with possibly any agent. This means repeatedly calling all agents with orders by there who number to simultaneously update their state. The traders values for currency and assets associated with the market are gotten and set during the course of the simulation through the extension. If the trader can’t afford to place an order, the extension will manage it by not placing the order and not holding currency or assets. When the fill completes, agents are updated with the appropriate amount of currency and/or assets. Any unfilled trades will return the original holdings of the agent.

More detailed explanation can be seen in the HTML guide for the auction extension, as well as some potential recipes for use.

3. Zero-Intelligence Economic ABM

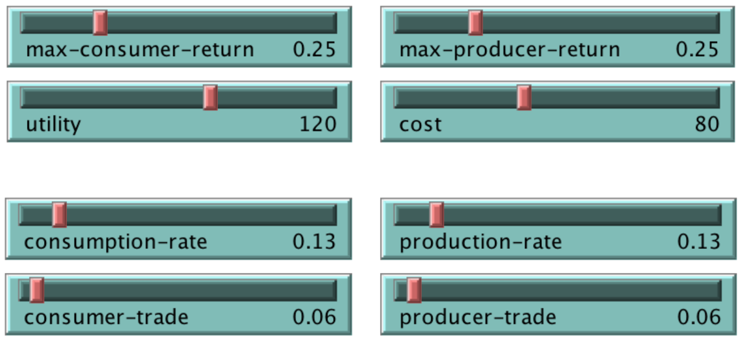
The model of interest has been inspired by the model of Zero-Intelligence traders. This model utilized agents with a random uniform pricing distribution to demonstrate allocative efficiency. The prices at which trades occurred was restricted to a narrow range by agents that were not acting in rational self-interest, but acted without any purpose or intent. (Gode and Sunder, 1993) The aspect that has been included in the implemented model from Gode and Sunder's paper is that prices are chosen not from any reasonable decision process, but as a random value within a range set by parameters. In the original paper, the experiment was run with 12 agents that repeatedly generated new bids and asks randomly from a distribution. The implemented model uses a slider to choose how many traders should be included. The distribution of prices in the paper's model is uniform and two contrasting models are run, one with a budget constraint, the other without a budget constraint. According to Google Scholar, this paper has been cited 1676 times and is the most recognizable work of either Gode or Sunder.

The model of Zero-Intelligence Traders in the paper is lacking in many ways that make it somewhat uninteresting. The budget constraint is not meaningful since there is no accounting mechanism, only a cost and redemption value to reduce the size of the distribution of uniform prices to sample from. Furthermore, prices at which to buy or sell are chosen from a distribution on each turn, making randomness an inherent structure of the model, not meaningful for the individual traders. The same effect could be achieved without agency. Finding overlap in these distributions may explain much of the pricing mechanisms in the original paper.

To make the model more interesting and useful, our model includes accounting functions that make order placement dependent on accumulated cash or assets, and the filling of orders a necessary condition for building account values. The equivalent of the original Zero-Intelligence model's cost and redemption value are respectively cost and utility in our model. In our new model, we will not choose prices each turn, but only a single price at which all trades for a particular agent occurs. This allows the agent to engage in profitable behavior through trading at agreeable prices. The agent stabilizes market forces at this price when trade occurs at the agent's price. If the agent get prices that are highly beneficial to its account, then account value will increase, driving prices further towards the agent's requested price. A stable requested price by each agent demonstrates an advantage of accumulating value based upon the interaction of agent demands in a complex adaptive system.

Our model seeks to build an idealized, symmetric model of behavior between consumers and producers in a market. The idea of symmetry underlies the ability of the model to verify accurate pricing mechanisms and accounting functions of trade. This symmetry can easily be broken by the user of the model because of a series of nearly symmetrical parameters designed specifically for the buy side (consumers) or the sell side (producers). Naturally, the model cannot be perfectly symmetrical by nature, since buying and selling are different functions. However, the differences can be seen as negligible. For example, buying 5 stock for 10 dollars each, can be seen as buying 50 dollars at a price of .1 stock each. These parameters allow control over the level symmetry:

- All agents generate a price from max-**x**-return and the value parameter (utility, cost)
- Utility and cost produce economic gains each tick for each agent
- **x**-rate is the rate of economic turnover in the simulation
- **x**-trade is the rate of trade in the simulation



max-consumer-return 0.25

max-producer-return 0.25

utility 120

cost 80

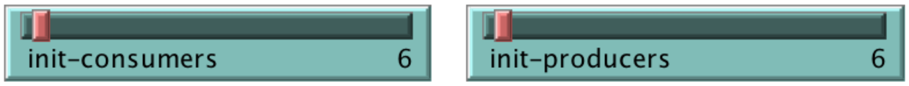
consumption-rate 0.13

production-rate 0.13

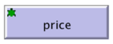
consumer-trade 0.06

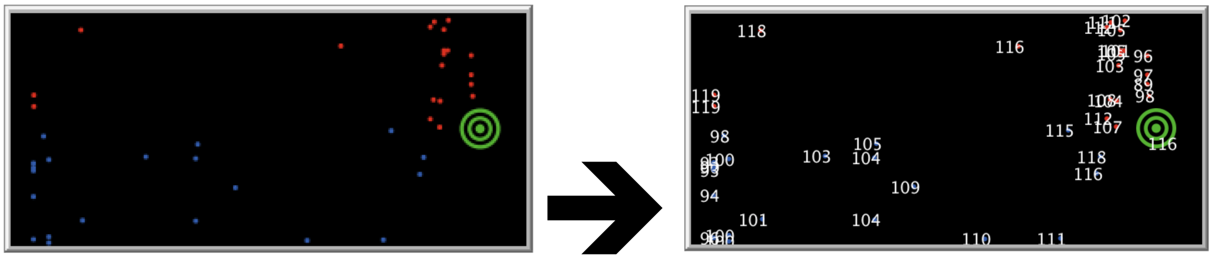
producer-trade 0.06

The model has the option to determine the profile of traders entering the market. The user is able to reproduce an environment that has 12 agents in the market by selecting 6 producers and 6 consumers in the init-produces and init-consumers tabs:

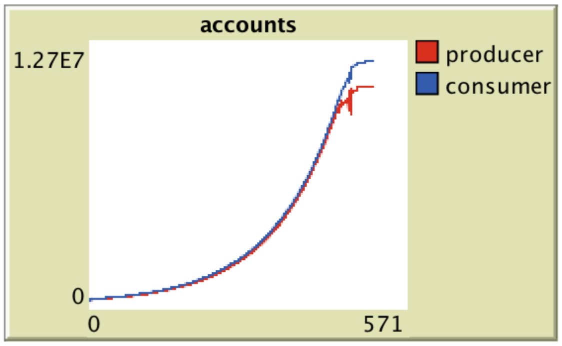


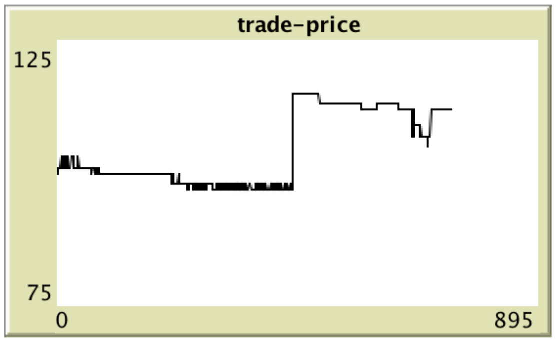
Being able to investigate alternative profiles of the market gives us a sense of which parameters are effectively comparable in their effects on exchange. For example, a large number of producers with low production-rate and producer-trade may aggregate identically to fewer producers with more production-rate and producer-trade. With enough runs of the model we could develop a function which would describe this exact relationship. This pattern can be seen in other models as well. The parameters of the Cooperation model (Wilensky, 1997), generally could be described by the costs and benefits of the agents. Parameters within the model could be modified in a complementary fashion to produce a similar outcomes.

The display of the model is built on a similar premise as the simple economy model. It shows total account values of traders by valuation of assets at the current trading price and the current holdings of cash. The further to the right the trader is, the greater this account value is. The market is displayed all the way on the right, but takes no actions. It will display the price if the price button is toggled. 

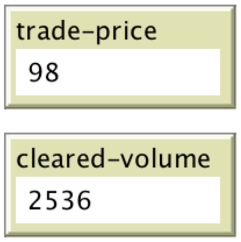


A better way to investigate the effects of the multi-agent system is to look at the data generated on aggregate account values and trade prices, which can be seen with the plots at the bottom of the simulation.

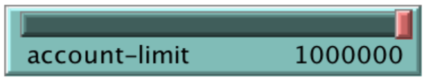




A monitor displays the current traded price as well as the cleared volume: The plot of accounts represent an exponential growth curve for all the traders in the market. It is limited by a parameter which prevents individual trading and economic functions if an account is too big:


trade-price
98

cleared-volume
2536


account-limit 1000000

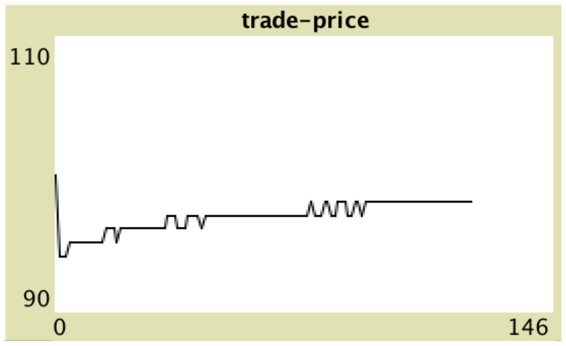
What has often been forecast as the natural trajectory of the economy is cumulative growth forever at an exponential rate. This is a hard aspect to negotiate in the modeling world. Allowing the model to grow forever causes computational errors. Creating a model that has significant account drawdowns may be validated in reality, but adds complexity which interferes with the purpose of the model. The purpose of the model is in analyzing price behavior based upon the growth of agents providing supply and demand to a market. Specifically, in an environment where prices are stable from the beginning of the model, and all traders have identical resources, we want to investigate how prices will find equilibrium.

Agents follow two major rules in the model. All traders first place an order to trade. Traders first ask “is my **account** under the **account-limit**”. If that is true, the trader’s rule is “I calculate a **trade-size** using **producer-trade** or **consumer-trade** (respectively) and place my order in the market, using my **price** (calculated at setup) and the **trade-size** for quantity”. Otherwise, traders do not trade. After this the market clears all orders of the trade. Traders relocate on the world map to show their respective account values. After this, producers and consumers perform their relative economic function if account in under the account limit. Each producer calculates a **production-cost** for the assets produced as **cash * production-rate**, and then calculate the **assets-produced** as **production-cost / cost**. The producer’s rule is “I will add **assets-produced** to my **asset**, and subtract **production-cost** from my **cash**”. Each consumer follows a nearly equal and opposite rule, calculating **assets-consumed** first with **asset * consumption-rate** and then calculating the gain as **asset-value** which is set to **assets-consumed * utility**. The consumer’s rule is “I will subtract **assets-consumed** from my **asset**, and add **asset-value** to my **cash**”.

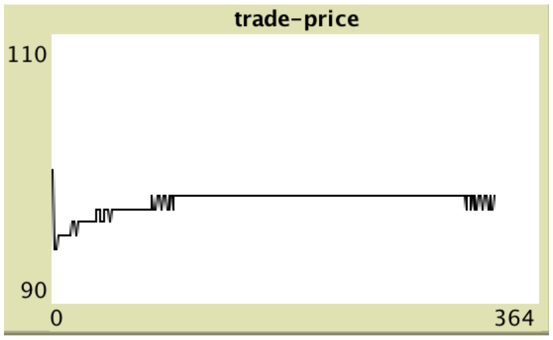
These fundamental economic activities provide a theory of trade as beneficial to parties involved, without something resembling this, trading must be a zero-sum game. Some very important concepts have been abstracted away through this. The **cost** and **utility** are well known. In most trading, the exchange occurs under the *expectation* of cost or utility. This would create the possibility of traders to lose money in the cycle of trading to economic activity. Furthermore, if the cost and utility are different for every trader, there will be more variance in prices from a genuine economic perspective. Including concepts of diminishing marginal returns is a more natural way to limit traders and production, rather than the account-limit. To compensate for expected price movements, each **volatility** and **trend** model patterns in both cost and utility.

4. Model Analysis

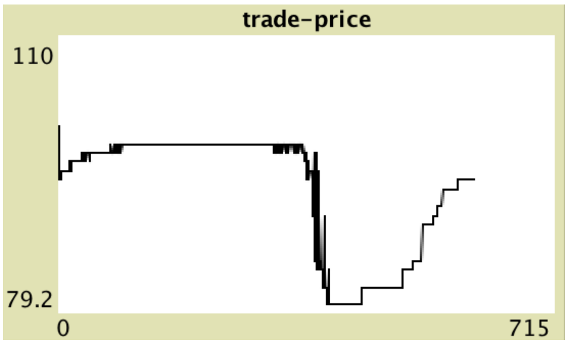
A typical price pattern that emerges in the model is an early swinging in prices as traders begin to build towards disequilibrium. Initially, all traders start with the same accounts and assets but start with different prices. This allows exchange to happen without regard to optimal prices, but rather just a function of how many traders land across the trade price. The trade price initially will split the traders in half. After a couple of cycles (ticks) traders will build accounts based on who has gotten filled and at what price. A symmetric initial market may look like this:



Eventually, this market evolves into an equilibrium price. The equilibrium price breaks down through a natural emergent process. Though an equilibrium exists in price, accounts can tip the balance:



Account imbalance can be significant when coupled with the price profile of traders. Economic rules cause divergence when traders are filled irregularly. Significant effects are seen when traders exit:



The emergence of this trading pattern, and the resolution that prices bounce back explains real world patterns. In financial markets, prices are held in place by large players before fast and strong moves.

5. Discussion

This model of economic and financial interactions to demonstrate the growth of markets and partial economies through mutually beneficial trade, demonstrated by efficient auctions can explain the interactions of supply and demand from individual agents on individual agents. Though trade has been around as long as humanity has, it has not always resulted in increased welfare. Only efficient trade has evolved into positive sum games due to the players requiring growth through exchange. This can be seen as a function of the proper mechanisms employed to induce trade by the proper beneficiaries over time. This project focuses on building and utilizing this type of auction mechanism to demonstrate this kind of trade.

In real world markets, traders refer to large players who stabilize prices as "paper" (In reference to large banks). Paper is indifferent to much of the effects of price movement. In this consumer/producer model, the economic functions are designed to mirror the activities of risk-free returns. Though this activity of removing players from the market when they get too large is induced, the emergent phenomenon that arises when a large player disappears from a market is unintentionally well modeled by the Zero-Intelligence Economic Trade model. More to the point, this happens in markets constantly. The supply and demand components of markets are made up mostly of gaps in timing of when agents are capable of going to market. This is the purpose of pure traders, to create supply and demand that fills gaps in time, place, or network. An auction provides an environment for agents to optimize allocation, rather than pure negotiations, which can only manage these gaps partially. The behavior of traders in this type of environment is well modeled as a complex adaptive system of financial exchange.

References

1. *Building the Santa Fe Stock Market,* Blake LeBaron, Brandeis University. 2002
2. *Allocative Efficiency of Markets with Zero Intelligence Traders: Market as a Partial Substitute for Individual Rationality.* Dhananjay K. Gode; Shyam Sunder. The Journal of Political Economy. Vol 101, No. 1 Feb., 1993.
3. *Fundamentals of Multiagent Systems with NetLogo Examples* Jose M Vidal. 2010
4. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*, Michael P. Wellman, Amy Greenwald, Peter Stone. 2007 MIT press
5. *NetLogo: http://ccl.northwestern.edu/netlogo/.* Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Uri Wilensky. 1997
6. *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with Netlogo.* Uri Wilensky and William Rand. 2015