Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments

Dave Cliff*
School of Cognitive and Computing Sciences
University of Sussex
BRIGHTON BN1 9QH, U.K.
davec@cogs.susx.ac.uk

June 1997

Abstract

This report describes simple mechanisms that allow autonomous software agents to engage in bargaining behaviors in market-based environments. Groups of agents with such mechanisms could be used in applications including market-based control, internet commerce, and economic modelling. After an introductory discussion of the rationale for this work, and a brief overview of key concepts from economics, work in market-based control is reviewed to highlight the need for bargaining agents. Following this, the early experimental economics work of Smith (1962) and the recent results of Gode and Sunder (1993) are described. Gode and Sunder's work, using "zero-intelligence" (ZI) traders that act randomly within a structured market, appears to imply that convergence to the theoretical equilibrium price is determined more by market structure than by the intelligence of the traders in that market: if this is true, developing mechanisms for bargaining agents is of very limited relevance. However, it is demonstrated here that the average transaction prices of ZI traders can vary significantly from the theoretical equilibrium level when supply and demand are asymmetric, and that the degree of difference from equilibrium is predictable from *a priori* statistical analysis. In this sense, it is shown here that Gode and Sunder's results are artefacts of their experimental regime. Following this, 'zero-intelligence-plus' (ZIP) traders are introduced: like ZI traders, these simple agents make stochastic bids. Unlike ZI traders, they employ an elementary form of machine learning. Groups of ZIP traders interacting in experimental markets similar to those used by Smith (1962) and Gode and Sunder (1993) are demonstrated, and it is shown that the performance of ZIP traders is significantly closer to the human data than is the performance of Gode and Sunder's ZI traders.

*This document reports on work done during February to September 1996, while the author held a Visiting Academic post at Hewlett-Packard Laboratories Bristol, Filton Road, Bristol BS12 6QZ, U.K.

This page is intentionally blank

Contents

List of Figures

1 Introduction

Developments in computer science and engineering over the last decade have seen a shift of emphasis from *centralized* to *decentralized* (or *distributed*) systems. In distributed computing systems, the problem of "von Neuman bottle-neck", i.e. having all commands and data pass through a single central processing unit, is avoided by employing multiple processors. The level of complexity of the individual processors may vary from simple processors with a small amount of local memory, as in the "Connection Machine" (Hillis, 1985), up to complex systems involving hundreds or thousands of high-performance workstations connected on a network. In the limit, the entire global internet could be considered as a single massively parallel distributed computer system.

This shift has led some researchers to develop new metaphors for thinking about computing. Bernardo Huberman, writing in the introduction to a landmark collection of papers discussing such systems (Huberman, 1988a), states:

> "A new form of computation is emerging. Propelled by advances in software design and increasing connectivity, distributed computational systems are acquiring characteristics reminiscent of social and biological organizations. These open systems, self-regulating entities which in their overall behavior are quite different from conventional computers, engage in asynchronous computation of very complex tasks, while their agents spawn processes in other machines whose total specification is unknown to them. These agents also make local decisions based both on imperfect knowledge about the system and on information which at times is inconsistent and delayed. They thus become a community of concurrent processes which, in their interactions, strategies, and competition for resources, behave like whole ecologies." Huberman (1988b, p.1)

In many computational ecologies, the allocation of (or competition for) scarce resources presents a serious issue. Scarce resources include processor time, storage (either primary, such as RAM-space; or secondary, such as disk-space), and network bandwidth. If no limits are put on the use of these resources, greedy or careless individuals can soon consume so much that the overall performance of the system suffers significantly.

The allocation of scarce resources is a topic that has long been studied in economics. In brief, if the quantity of a resource demanded by consumers in a market is greater than the quantity supplied, the price of the resource rises, which can both reduce the quantity demanded (because some consumers can no longer afford it) and also increase the quantity supplied (because some suppliers may be more interested in selling at higher prices). Similarly, when the quantity supplied is greater than the quantity demanded, the price falls, which may reduce the quantity supplied and increase the quantity demanded. Thus, according to classical economic theories of markets, the price approaches an *equilibrium* value where the quantity demanded matches the quantity supplied.

Advocates of free-market economics claim that the actions of groups of individuals, engaging in simple trading interactions driven by self-interest, can result in good or optimal allocation of resources. Crucially, the market mechanism does this in a distributed fashion: there is no central control process; rather, the allocation of resources 'emerges' from the interactions of the buyers and sellers.

Thus, economics can act as a valuable source of terminology, inspiration, and metaphors for developing solutions to problems in distributed resource-allocation and control. This theme has been explored in detail in the growing research field known as *market-based control* (MBC).

In brief, the aim of MBC systems is that groups of software 'agents' or 'traders' interact within a market-like framework. In general, the inspiration from market economics comes in the form of division of the scarce resources into units of 'commodity' and the provision of a 'currency' that allows the agents to buy or sell the commodity. Some agents act as 'producers' or 'sellers' of the commodity (e.g., an agent may be assigned to a node or link in a network, charging for use of that resource), while others act as 'consumers' or 'buyers' (e.g., an agent may be assigned to a data-packet on a network, spending currency in order to route the packet from its source to its destination). In principle, when supply is greater than demand, the price of the commodity will fall; and when demand exceeds supply, the price rises. The aim then is that prices rise and fall, dynamically matching the quantity demanded to the quantity supplied.

One of the powerful arguments for adopting a MBC approach is the prospect of *automation* and *decentralisation* of control or resource-allocation. For the process to be automatic, it should devolve power from human operators (who have a tendency to be expensive and/or error-prone). Ideally, once the system is operational, human input should be reduced to a minimum. And for the process to be decentralised, there should be minimal reliance on central control mechanisms, processes, or databases (e.g., models of the entire network). Ideally, the system should not rely on the operation of any single critical component or sub-system. The failure of any one component should result in only a minor impairment, if any, to the overall behavior of the system; rather than a total collapse.

Yet, to the best of my knowledge, no current MBC systems are both automatic and distributed in the senses just described. In the applications published in the literature, there is a reliance either on centralised 'auctioneer' processes or on human intervention. In the case where a centralised auctioneer process is used, in addition to the brittleness caused by failure of the central process, there are issues of imposition of synchrony on what is fundamentally an asynchronous system, and the maintenance of the central auctioneer's database (i.e., its 'knowledge' of how many buyers and sellers there are, their interconnections, etc.).

For this reason at least, there is a need for computational 'bargaining' mechanisms that allow a software agent to decide what price to agree on for a deal (ie., for a transaction): sellers should not set a price so low that they make a loss, but should also beware of setting a price so high that no buyer is interested; buyers should not spend any more than they have to, but should not bid so low that they are unable to purchase the commodities they require. Clearly, the price announced by an agent (either a seller's offer-price or a buyer's bid-price) is likely to be influenced by the prices announced by other agents in the market.

But the potential use of such bargaining mechanisms would not be limited to MBC applications. There are at least two other significant application areas in which autonomous software agents with bargaining abilities could be profitably employed:

- **Internet Commerce:**

  The recent explosive increase of activity on the internet and world-wide-web, and in particular the announcement of secure transaction methods for transfer of funds, offers the potential for exploiting internet-based commerce. Although there are many possible methods by which goods and services can be advertised and sold on the internet, one possibility is the use of software agents that autonomously seek out and purchase items on behalf of a human user, entering into bargaining (or "haggling") interactions with agents representing the sellers of the desired items. The user would, presumably, indicate preferences such as price-range, desirable features, et cetera, and the buyer-agent would then traverse the web, seeking information from vendors' web-sites and interacting with each vendor's seller-agent, trying to strike a good deal. While the application of such techniques to the purchase of domestic goods such as music systems or kitchen appliances may be somewhat fanciful, a more realistic possibility might be the use of autonomous bargaining agents to automate auctions in financial asset markets: in particular, many international derivative markets (where options and futures are traded) are based on open-outcry 'trading pits' where human traders announce offer and bid prices to the other traders present, without a mediating centralised auctioneer. If automated, such markets could (in principle) be made more efficient because new information could be distributed and assimilated at electronic speeds, and there should be less susceptibility to the effects of crowd-psychology, such as fluctuations in price caused by unfounded rumours.

- **Positive Economic Models:**

  There is a distinction within economics between two branches of the field: *positive* and *normative*. Normative economics deals with subjective recommendations based on personal value judgements, while positive economics aims at providing 'objective' or 'scientific' explanations of economic phenomena (Begg, Fischer, & Dornbusch, 1994, p.11). Much work in positive economics explores theoretical models based on axiomatic assumptions concerning the behavior of individual agents, such as humans or firms in a particular market. Often, the primary behavior exhibited by each agent is a decision between two or more possible actions. Whether the results of the action are beneficial to the agent can depend on the actions of other agents within the system, thereby inviting game-theoretic analysis.[1]

  The degree of *rationality* expected of economic agents has been a topic of considerable debate. A large body of economic modelling research is based on the so-called *rational expectations* hypothesis; the belief that agents base their behavior on predictions which are free of systematic errors: "Nobody can predict the future with perfect foresight because unforeseen, random happenings are bound to occur. However, someone with rational expectations will construct their expectations so that on average they are correct; that is, they will be wrong only because of random, non-systematic errors." (Bannock, Baxter, & Davis, 1992, p.360). Although many models based on the assumption of perfect rationality have good predictive power, some economists have argued that real agents are unlikely to exhibit optimal rational behavior, even if that is their intention: they may not have the necessary information or cognitive power to do so; in which case they are said to exhibit *bounded rationality.* The behavior of bounded-rationality agents is better characterised not as optimizing some variable (i.e., attempting to achieve its maximum possible value), but rather as *satisficing* (i.e., attempting to keep the variable above some minimum level) (Bannock et al., 1992, p.380).

  Bounded rationality introduces extra constraints on any modelling process, and the accuracy of predictions from rational-expectations models may need to be verified by empirical means. One possible empirical method is to use historical data, but in many cases the data may not be available in sufficient quantities or with sufficient accuracy to validate the model. An alternative lies in so-called *experimental economics*, where laboratory-style experiments are conducted with groups of human subjects interacting in controlled market conditions, with significant variables being recorded for subsequent analysis and comparison. Yet such experiments can be costly and time-consuming, because of their reliance on human subjects.

  A complementary approach was described by Brian Arthur, who discussed the idea of designing artificial agents (software automata) that behave like human economic agents.

[1]For introductions to game theory, see e.g. Fudenberg and Tirole (1991) and Heap and Varoufakis (1995).

> Arthur (1993) explored the use of a simple machine learning algorithm to develop artificial agents which could be calibrated against human learning data from psychology experiments. The use of artificial agents in economic modelling offers two benefits. First, the costs associated with using humans are eliminated. Second, the use of computer simulations forces a degree of mechanistic rigour, and thus helps clarify what information and/or cognitive mechanisms are necessary and sufficient to produce the behaviors of interest. In this sense, the algorithms or programs that specify the behavior of the artificial agents can be viewed as 'theories' concerning the generation of comparable behaviors in real agents.

So, in summary, artificial agents capable of bargaining behavior could find uses in market-based control, internet commerce, and economic modelling.

This paper reports on research and development work aimed at identifying minimal mechanisms that endow autonomous software agents with bargaining behaviors appropriate to market-based environments. The emphasis on minimalism comes not only from a desire for computational efficiency (important if hundreds or thousands of such agents are active on a network), but also in a speculative attempt at sketching the minimum mechanistic complexity necessary and sufficient for explaining human bargaining behaviors in simple market environments. The contents of the rest of this paper are as follows.

Section 2 presents a brief tutorial review of relevant concepts and terminology from economics. Section 3 then reviews work in MBC, highlighting the reliance on centralised auctioneer processes and human intervention, which initially motivated the research described here.

Following this, in Section 4, there are summaries of two bodies of work in experimental economics: the first is early work by Vernon Smith, and the other is more recent work by Dhananjay Gode and Shyam Sunder. Smith's experiments were some of the first to demonstrate that small groups of human traders could rapidly approach the theoretical equilibrium price predicted by rational-expectations models. Meanwhile, Gode and Sunder's results appear to indicate that groups of so-called 'zero-intelligence' trading agents, that simply announce random prices for bids and offers, can give results that are remarkably similar to the performance of human traders. The implication drawn from their work is that the structure of the market plays a large part in determining the observable behavior of the agents in that market, and the intelligence (or lack of it) of the agents in the market is secondary. If this is true, the relevance of developing computational mechanisms for bargaining agents is highly questionable.

However, in Section 5, I present analytic arguments that Gode and Sunder's results are affected by experimental artefacts, and that zero-intelligence traders only approach the theoretical equilibrium price when the experimental market is structured appropriately. The analytic arguments are supported by results from simulation studies where zero-intelligence traders are employed in market environments similar to those that Smith used with human subjects. These results demonstrate that more than zero intelligence is required to exhibit human-like equilibration.

Then Section 6 reports on the development of 'zero-intelligence-plus' (ZIP) traders that, in the spirit of Gode and Sunder's work, are stochastic traders with minimal intelligence, but incorporate elementary machine learning techniques to alter their behavior on the basis of experience. It is demonstrated that in experimental conditions comparable to those used by Smith, where Gode and Sunder's zero-intelligence traders either fail to produce human-like tendency to equilibrium or simply cannot be used, the behavior of the ZIP traders is comparable to that of human subjects.

Related work is described in Section 6.5, and possible further developments of this research are discussed in Section 6.6, before offering conclusions in Section 7. Finally, the C-code for the programs used in the simulations is presented in Appendix A, and sample input and output files are given in Appendix B.

This page is intentionally blank

2 Background Economics

2.1 Basics

The allocation of scarce resources is a topic that has long been studied in economics. Indeed, the notion of a *scarce resource* has a precise definition: it is any resource "...for which the demand at a zero price would exceed the available supply" (Begg et al., 1994, p.4). The area of economics known as *microeconomics* is primarily concerned with the allocation of scarce resources: as Gravelle and Rees (1992, p.1) state, "Microeconomics is a set of theories with one aim: to help us gain an understanding of the process by which scarce resources are allocated among alternative uses, and of the role of prices and markets in this process."

More precisely, a *market* can be defined as "...a set of arrangements by which buyers and sellers are in contact to exchange goods or services" (Begg et al., 1994, p.32). The quantity of a commodity (good or service) that buyers are prepared to purchase at each possible price is referred to as *demand*, and the quantity of a commodity that sellers are prepared to sell at each possible price is referred to as *supply*. In general, the greater the price of a commodity, fewer buyers will be inclined to make a purchase, and so the quantity demanded reduces: thus, if we plot price as a function of quantity, the *demand curve* slopes downward. In contrast, the greater the price of a commodity, the more sellers are inclined to sell, and so the quantity supplied increases: on a plot of price as a function of quantity, the *supply curve* slopes upwards. From these considerations, it is clear that at high prices the quantity supplied may exceed the quantity demanded (i.e., there is a surplus, or *excess supply*), and at low prices the reverse may be true (giving a shortage, or *excess demand*). But, at some intermediate price, the quantity demanded is equal to the quantity supplied: this is the *equilibrium price*, which 'clears the market': graphically, the equilibrium price (and quantity) can be determined by the intersection of the supply and demand curves, as illustrated in Figure 1. Throughout the rest of this document, where mathematical notation is used, the equilibrium price will be represented by P_0 and the equilibrium quantity by Q_0. The equilibrium quantity is also sometimes referred to as the "clearing quantity".

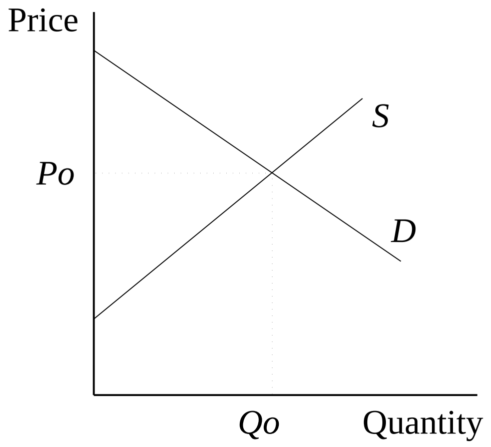

Figure 1: Simple illustration of supply and demand. The Supply Curve S slopes upwards and the Demand Curve D slopes downward. The two curves intersect at a point indicating the Equilibrium Price P_0 and the Equilibrium (Clearing) Quantity Q_0.

In theory at least, markets are *self-correcting*: if the supply and demand schedules remain fixed, the transaction prices in the market will tend toward the equilibrium value. At prices below equilibrium, there is excess demand (or a *sellers' market*): the suppliers can then choose only to sell to the highest-bidding buyers, and so the buyers have an incentive to bid higher prices, thereby raising the market price towards the equilibrium value. At prices above equilibrium, there is excess supply (or a *buyers' market*) and buyers can then choose only to buy from the sellers with the cheapest offers, so sellers have an incentive to cut their offer prices, thus lowering the price towards equilibrium. At the equilibrium price, neither buyers nor sellers have any incentive to change their prices, and so the system stabilises. In this sense, the actions of a group individuals in a market, each pursuing their own interests, can give rise to *price determination*, or *equilibration*, where the market price is the equilibrium price and so the quantity demanded matches the quantity supplied. Because the equilibrium is a result of price-competition between the agents in the market, it is often referred to as a *competitive equilibrium*. For an example of a formal definition of competitive equilibrium, see Roth and Oliveira Sotomayor (1990, p.209).

Such market mechanisms, it is argued, can give efficient (or perhaps optimal) allocation of resources without centralised control or external intervention (e.g., by government regulation). A common ideal of allocative efficiency is *Pareto efficiency*: an allocation of resources is Pareto-efficient if no-one can be made better-off without someone else being made worse-off. If there is no external intervention, the system is said to be a *free market*. Free markets can, according to their proponents, give rise to 'emergent' behavior (competitive equilibrium) of the whole group which is in the best interests of that group (Pareto-efficient), despite the fact that each agent is operating only to satisfy self-interest. The group appears to be led by an 'invisible hand', in the metaphor introduced by Adam Smith in his 1776 book *The Wealth of Nations*. There are, however, conditions in which free-markets fail to achieve an efficient allocation: see Begg et al. (1994, pp.264–265).

Conditions under which a market can attain equilibrium from a given set of initial conditions, and the nature of the approach to equilibrium, has been the subject of intense research in economics. The nineteenth century economist Marie Walras suggested a *tâtonnement* (groping) process that resembles an iterated auction: in the first round, buyers and sellers announce their prices. In subsequent rounds if there is excess supply the buyers and sellers reduce their prices, or raise them if there is excess demand, or leave them unaltered if supply matches demand, at which point actual transactions are permitted to take place (Bannock et al., 1992, p.417). More sophisticated models of equilibration include the *cobweb model*, where there is cyclical demand and supply with the response of suppliers to changes in price being subject to a delay, and the system dynamics converge on either a point attractor (at equilibrium), a periodic attractor around the equilibrium point, or diverge with increasing fluctuations in price and quantity (Bannock et al., 1992, pp.72–73). Many models of equilibration assume *perfect competition*, where homogeneous indivisible units of a commodity are traded by large numbers of buyers and sellers, none of whom are sufficiently large or powerful to have any impact on market price, all of whom are aiming to maximise profit, and none of whom incur any costs in entering or leaving the market (Bannock et al., 1992, p.325). The nature or organisation of some markets makes perfect competition unlikely or impossible, yet price determination can still occur. However, as the number of individuals in the market falls, the likelihood of collusion and the formation of rings or cartels increases: as the number of sellers is reduced, the market approaches an *oligopoly*, where the behavior of individual sellers in the market is highly dependent on the likely initiatives or responses of the other sellers: the limiting cases are the *monopoly* market (with only one seller), the *monopsony* market (with only one buyer), and the *bilateral monopoly* market (with one buyer and one seller). In all these cases, the actions of individual buyers or sellers can have a significant impact on the market price, depending on the organisation of the market.

2.2 Market Organisation

Market organisation concerns the regulations governing the information available to the agents in the market and the agents' *opportunity sets* (i.e., the possible actions they can perform). Together, these affect the method by which a market price is determined. In most markets, prices are determined via a particular style of *auction*. Colloquially, the word 'auction' is used to refer to arrangements where sellers of a commodity and a group of potential buyers of that item interact to agree a price. There are a large number of types of auction mechanism, and the literature on auctions indicates that different authors sometimes use the same name to refer to differing mechanisms. In brief: in an *ascending bid* or *English* auction, buyers make increasing bids for the sale item, withdrawing from the process as the price rises, until only one buyer remains. In a *descending offer* or *Dutch* auction, the seller offers the item at decreasing prices, and the first buyer to agree to take the item at the current price gets the deal. In a *sealed-bid* auction, all buyers submit a single bid-price 'sealed' to prevent observation by the competing buyers. Some sealed-bid auctions are *iterated*, where information about the bids (such as the full list of prices, or just the highest price) is made public after each round of bidding. The alternative is a *single-round* auction. In single-round sealed-bid auctions the highest-bidding buyer usually purchases the item: in some variants at the highest bid price (first-price auctions); in others at the price bid by the second-highest-bidding buyer (the so-called second-price sealed-bid or *Vickrey* auction, for which there are game-theoretic proofs that honesty is an optimal strategy). For discussions of different auction market organisations and a useful bibliography, see Agorics, Inc. (1997)

One of the simplest forms of market mechanism for matching supply and demand is the *call auction*: a central impartial auctioneer collects bids from all buyers and offers from all sellers; the array of bids is used to determined the market demand curve, and the offers determine the supply curve; the intersection of the two curves gives the market-clearing (equilibrium) price, and all possible trades clear simultaneously at that price (Satterthwaite & Williams, 1992, p.92). Call markets are employed in a number of international financial exchanges, especially where there are periods or conditions of low volume or low liquidity.

One particular style of auction has received significant attention in the literature: the *continuous double auction*, where a group of buyers and a group of sellers simultaneously and asynchronously announce bids and offers: at any time, a seller is free to accept the bid of a buyer, and a buyer is free to accept the offer of a seller. The continuous double auction (CDA) therefore resembles a parallel integration of English and Dutch auction styles. One practical reason for the interest in CDAs is that they have for many years been the basis of trading in major international financial markets (such as the London and New York stock exchanges): originally as open-outcry oral auctions taking place on the trading floor of the exchange, and latterly as electronic auctions taking place in the cyberspace of city-wide networks of computerised dealing rooms. Theoretical interest is motivated by the fact that CDAs are often very fast and efficient, despite (or possibly because of) the volumes of information exchanged:

> "...markets organized under double-auction trading rules appear to generate competitive outcomes more quickly and reliably than markets organized under any alternative set of trading rules. For this reason, double-auction markets have been frequently investigated as a standard against which the performance of other institutions is evaluated." Davis and Holt (1993, p.126).

This interest has resulted in a research literature discussing CDAs (see, e.g., Friedman and Rust (1992) and Davis and Holt (1993, pp.125–172) for reviews) The motivation for such work comes from a desire to better understand *how* the organisation of the market, and the behavior of the traders in that market, affects the speed and efficiency of the market. This is summarised neatly by Rust, Miller, and Palmer (1992, p.156):

> "Although the textbook 'supply equals demand' model may provide a good prediction of *closing* prices and quantities in [double auction] markets, it fails to explain the dynamics by which this happens. A more sophisticated theory is required to show how the trading process aggregates traders' dispersed information, driving the market towards [competitive equilibrium]. The essence of the problem was clearly stated by Friederik Hayek nearly 50 years ago:
>
> > "The problem is in no way solved if we can show that all facts, *if* they were known to a single mind, would uniquely determine the solution; instead we must show how a solution is produced by the interactions of people each of whom possesses only partial knowledge. To assume all the knowledge to be given to a single mind in the same manner in which we assume it to be given to us as the explaining economists is to assume the problem away and to disregard everything that is important and significant in the real world." (Hayek, *Amer. Econ. Rev.* 35(4):p.530, 1945).

Much of the literature deals with auctions where there is no inherent asymmetry between buyers and sellers: buyers can, in principle at least, become sellers (i.e., can buy a stock of items and then sell them on the same market they bought from) and vice versa. This flexibility helps equilibration. For example, when prices are low through excess supply, sellers might cease to offer a commodity, preferring to buy it from the market: in doing so, supply is reduced and demand is increased; and the market moves closer to equilibrium. However, in some types of market, this flexibility is not possible because of inherent asymmetries: a transatlantic pilot dissatisfied by low pay (i.e., income from selling her services) cannot become an international airline (i.e., a buyer of pilot services); a doctor cannot become a hospital; and a professor cannot become a university. In such cases, known as *two-sided matching* markets, special game-theoretical analyses and algorithms have been developed: see e.g., Roth and Oliveira Sotomayor (1990).

Finally, because it will be relevant later in this document, one embellishment of the continuous double auction needs to be mentioned here: some CDAs employ an *improvement* rule, introduced to speed the auction process, that requires that each bid or offer leading to a transaction must be an improvement on the previous bid or offer (i.e., a higher bid or a lower offer) until a transaction is agreed, at which point the process is reinitialised by the first bid and offer for the next item or lot: such a rule is used at the New York Stock Exchange (NYSE), and is commonly referred to as the NYSE rule (e.g., Easley and Ledyard (1992, p.84) and Cason and Friedman (1992, p.258)).

3 Market-Based Control

The theme of using ideas from microeconomics in controlling computational ecologies was explored in depth by Mark Miller and Eric Drexler in three papers proposing the development of what they refer to as *Agoric Systems*: distributed computer systems, possibly involving multiple owners or vendors, where scarce resources are allocated using techniques inspired directly by the operation of free-market economies (Miller & Drexler, 1988a, 1988b; Drexler & Miller, 1988). Their work represents early methodological arguments for a growing field that is now more widely known as *market-based control* (MBC).

In a recent collection of papers describing studies in MBC edited by Clearwater (1996), while there are some systems which are decentralised but not fully automatic, and other systems which are automatic but centralised, there is an absence of systems which are both automatic and decentralised. Enumerated below are brief summaries of some of the papers in Clearwater's collection that describe actual applications rather than theoretical models.

3.1 Network Bandwidth

Miller, Krieger, Hardy, Hibbert, and Tribble (1996) discuss a system for automated auction of ATM (asynchronous transfer mode) network bandwidth. The intention is that bandwidth (a scarce resource) is traded as a commodity by a community of software agents. In times of low network usage, high-bandwidth network connections may have a low 'price': as usage increases, so the demand for bandwidth increases and the price of the resource rises. Once demand has increased, users of the system have to make a simple decision between maintaining the previous level of expenditure (and consequently accepting reduced quality of service) or maintaining the prior quality of service (at a higher price). The system is sophisticated, and its components include mechanisms of banking and currency, bidding agents, auctioneers, delivery agents, and application and user interfaces. However, as is implied in Miller et al. (1996) and made explicit in a technical report published by the developers (Agorics, Inc., 1996, p.21), the system relies on a centralised auctioneer process, known as NetAuctioneer.

> "The current implementation of NetAuctioneer is unrealistic in two regards. First, it is a single centralized entity which requires a global model of the network, rather than a distributed network of auctioneers each of whom have local knowledge only of parts of the net. Second, it computes both allocations and prices by combinatorial search with exponential cost, rather than approximating ideal results in exchange for a reasonable computational burden." Agorics, Inc. (1996, p.21).

The need for a global model of the network in NetAuctioneer may incur high maintenance costs in large or complex networks: presumably, any change to the network structure (including sudden failure of nodes or links) has to be reflected in the model for the auctioneer process to operate successfully. Moreover, there is the manifest danger that the entire bandwidth allocation system will collapse if the machine running the NetAuctioneer process fails.

3.2 Memory Allocation

Harty and Cheriton (1996) describe the application of a market approach to memory allocation in a computer operating system. In their system, there is no increase in the price of memory in response to high demand. Rather, they use a tiered pricing system that allows the requestor to indicate the urgency or priority of a request. (Harty & Cheriton, 1996, p.152). When demand exceeds supply, the memory allocation scheme gives initial priority to those applications which have sufficient money to pay for their requested memory: when there is more than one such application, "...the allocation scheme gives priority to those applications that request an amount of space-time that is less than or equal to their 'fair share'." (Harty & Cheriton, 1996, p.153). Again, this notion of 'fair share' requires a *global* view of the system: the 'fair share' of any one application can only be determined by reference to the needs and expenditure of all other applications.

3.3 Air Conditioning

Clearwater, Costanza, Dixon, and Schroeder (1996) demonstrate the use of market-based techniques for control of air-conditioning ventilation and temperature. In their system, used to manage the air conditioning of 53 offices at the Xerox Palo Alto Research Center (PARC), software agents represent individual temperature controllers bid to buy or sell conditioned air. Unlike conventional building energy management systems, the Xerox system can take account of the interactions and connectivity between offices, and results indicate that the market-based system gives better distribution of temperatures and uses fewer resources. Again, a central computerized auctioneer process is employed:

> "A central auctioneer collects the bids and computes the supply and demand curves and sets the [equilibrium] price for the auction. Agents whose bids were not too high or low have their trade consummated. ... Buyers buy only if the trade [equilibrium] price is at or below their buy offer. Sellers only sell if the trade [equilibrium] price is at or above their sell offer. All other traders must wait for another auction to attempt having their bid consummated." (Clearwater et al., 1996, pp.256–257).

3.4 Pollution Regulation

Marron and Bartels (1996) recount their experiences in developing computer-assisted auctions for the allocation of tradable pollution permits. A pollution permit gives a firm the right to emit some specified amount of pollution over a specified period of time. By giving firms an initial allocation of permits and then allowing them to trade them in an active free market, each firm acts as a participant in a decentralized decision-making process. If the cost of reducing pollution by some amount is less than the market-price of a permit to produce that amount of pollution, it is profitable for a firm to reduce its pollution output and sell the corresponding permit. Firms that are unable to reduce pollution can buy extra permits from seller-firms, to avoid punitive financial penalties imposed on firms that produce pollution without a permit. The process is decentralized in the sense that there is a reduced emphasis on government regulations that specify detailed standards covering equipment, operations, and so on (Marron & Bartels, 1996, p.275). However, once more, a centralized computerized auction mechanism is employed (Marron & Bartels, 1996, p.283).

3.5 Job-Shop Scheduling

Finally, Baker (1996) reports on the development of a fully distributed computer architecture for factory control, based on a market-driven contract net. In this system, a network connects agents that each control one or more aspects of a manufacturing system, such as particular machines, inventory storage, material-handling, and so on. Also connected to the network are 'sales' agents that allow customers to request a product. When a product request is received by a sales agent, the agent issues a 'request for bid' (RFB) on the network. Any agent that receives the RFB and is capable of delivering that product replies to the sales agent with a WILL-BID message and simultaneously issues RFBs for all the components of the product. Agents capable of supplying the components issue WILL-BID messages to the finished-product agent and issues RFBs for the materials or sub-components necessary to manufacture the components. This process continues forward through the network until RFBs are received by agents responsible for purchase or stock-control of raw materials: those agents then issue a BID to all agents that requested materials from them. Agents receive the BIDs, combine them with any other BIDs requested, and calculate their own BID. This process of bid-collation and calculation proceeds back through the network until bids are received by the sales agent that issued the initial RFB. This agent combines the bids and presents them to the user who made the request. Because the network will typically contain multiple paths from the sales agent through different machines or processes to the material supply, so the final bid is a multi-dimensional item of data: Baker (1996, p.190) illustrates the bid to a customer as a 3-dimensional surface relating lot-size (number of units), delivery-time (weeks), and cost per unit. It is the responsibility of the customer to choose a combination of lot-size and delivery-time that gives an acceptable unit cost.

In Baker's system, the prices attached to bids are determined by the convolution of a number of functions relating lot-size, production-time, and cost: examples include employee pay-rate as a function of time, time-required as a function of lot-size, cost per unit as a function of delivery time, and current capacity status (available or unavailable) as a function of time (Baker, 1996, p.192). Following the convolution of these functions, a 'inventory smoothing' operation is applied that accounts for the possible savings incurred by producing a product before it is needed (e.g., at times of low demand) and then holding it in the inventory.

Although individual agents within the system may have varying parameters for these functions, additional work is required to determine the appropriate functions and parameters, or to alter them as a consequence of changing circumstances. In the case-study reported by Baker, it is assumed that information regarding whether an agent can make a particular sub-component, and the amount of set-up and processing time required to make it, are determined by a table-lookup procedure. As Baker (1996, p.194) honestly notes, "...this is a major assumption which may not be true". Presumably, prior empirical or theoretical work is required to determine the functions and parameters for each agent in order to fill the entries in the lookup tables. Thus, although Baker's system has a decentralized architecture, the collection of functions and parameters, or the entries in the lookup table, for all the agents in the system is effectively a global model (albeit one that is distributed across the system at implementation time). There is a requirement for *a priori* knowledge to be incorporated and for manual intervention when the machine an agent represents is modified or fails.

3.6 Summary

From this brief review, it is clear that although decentralization, parallel distributed processing, and self-organisation are strong motivating factors for the development of MBC systems, the applications reviewed above are all lacking in some respect: centralized auction processes, similar to call markets, were used by Miller et al. (1996), Harty and Cheriton (1996), Clearwater et al. (1996), and Marron and Bartels (1996); while Baker's (1996) work, although clearly inspired by real markets, employs no auction mechanism because the human user is presented with all options to choose from.

Yet real markets operate efficiently without centralised auctioneers. The continuous double auction is an attractive market structure: it is fundamentally asynchronous, and examples of distributed continuous double auctions pervade real markets. In a computational ecology or MBC system, groups of locally-connected agents could engage in continuous double auctions to determine the price for commodities, eliminating the need for pre-compiled price tables or functions based on *a priori* knowledge. Price changes might percolate through the entire network, or there might be a spatially heterogeneous price distribution (just as the price of hot-dogs in London is determined, in part, by a double-auction haggling process that has little impact on the price of hot-dogs in Brighton, 60 miles away: hungry tourists in London might get a discount by arguing that cheaper hot-dogs are available two streets away, but comments that hot-dogs are half the price in Brighton are unlikely to have much effect.)

Despite the attractions of MBC, it is important to note that there are a number of theoretical studies that raise difficult issues. In particular, there are indications that the dynamics of markets populated by simple traders acting purely to serve their own self-interest may converge to stable but highly sub-optimal equilibria (e.g., Glance & Huberman, 1994), or may exhibit complex chaotic and hyperchaotic dynamics, (e.g., Glance & Huberman, 1992; Thomsen, Mosekilde, & Sterman, 1992). The extent to which such theoretical models are applicable to real market systems is a matter for current empirical research. Nevertheless, attempts at emulating the success of human double-auction markets could be helped by a better understanding of those human markets. There are at least two relevant issues: what are the typical dynamics of human double-auction markets, and to what extent are those dynamics dependent on the market structure and the level of intelligence of the traders in the market? Work in experimental economics that addresses these issues is described in the next section.

4 Experimental Economics

The use of 'laboratory methods' in economics, conducting controlled experiments to test theoretical hypotheses or predictions, has been of interest since at least the 1930's: for historical reviews, see Davis and Holt (1993, pp.5–9) and Roth (1995, pp.4–21). In a typical simple experiment, a group of human subjects are each given the means to buy one unit of an arbitrary commodity; while another group are each given one unit of the commodity to sell. Each buyer will be a given a maximum limit price to pay for a unit of the commodity, and each seller will be given a minimum limit price below which the unit should not be sold. Typically, different buyers will be given different limit prices, as will different sellers. The manner in which limit prices are allocated determines the supply and demand for the experiment. The subjects are then allowed to buy and sell within a particular market mechanism: in the early experiments, the markets were experimentally controlled open-outcry trading pits, but the vast majority of recent work has required the subjects to communicate via a computer network, which eases the control of free parameters and the recording of data. Theoretical hypotheses concerning the effects of market-structure on the dynamics of the market can be tested and their implications explored by varying parameters such as supply and demand, what trader actions are permissible, or the amount and quality of information available to each trader. Factors of interest may include the nature of the approach of observed transaction-prices towards the theoretical equilibrium price, the stability at equilibrium, the amount of potentially-available profit that is extracted from the market by the sellers, etc.

Davis and Holt (1993) wrote a comprehensive text covering the major areas of experimental economics, while Kagel and Roth (1995) edited a significant collection of critical surveys of the field. Methods and results from two key papers in the field are summarised below. Section 4.1 summarises the first paper on experimental economics published by Smith, who helped establish the field and has since continued to be a prominent researcher: for a brief overview of his work, see Smith and Williams (1992), and for more complete details see his collection of papers spanning 30 years of research (Smith, 1992). Section 4.2 summarises work done by Gode and Sunder on 'zero-intelligence' (ZI) traders. This work has been cited approvingly in texts on experimental economics (e.g., Friedman and Rust (1992, p.*xxiii*), Friedman (1992, p.19), Rust et al. (1992, pp.160–161, 175), Bollerslev and Domowitz (1992, pp.230–231), Cason and Friedman (1992, pp.253, 258), Kagel and Vogt (1992, pp.292, 294), Davis and Holt (1993, p.132), Roth (1995, pp.52–55, 80–81), Holt (1995, p.370), Sunder (1995, p.475), Kagel (1995, pp.570, 580), and Camerer (1995, p.674)), and has even been discussed in a recent book on the philosophy of cognitive science (Clark, 1997, pp.183–184).

However, it is argued in Section 5 that Gode and Sunder's results are artefactual, and this provides a strong motivation for the development of artificial trading agents with (slightly) more than zero intelligence, as described in Section 6. Smith's work is reviewed here both as a means of introducing terminology and typical methods of experiment and analysis, and also to establish a more varied set of experimental scenarios than were used by Gode and Sunder: the arguments in Section 5 are supported by considering the performance of Gode and Sunder's ZI traders in experimental settings similar to those used by Smith.

4.1 Smith's Experimental Study of Market Behavior

Smith (1962) reports on experiments performed over a six-year period starting in 1955. All of the experimental regimes are similar to that described above: a group of human subjects are divided into a sub-group of sellers (each with an entitlement to sell one or more units of a commodity at a price no lower than their specified limit price) and a group of buyers (each with an entitlement to buy one or more units of the commodity at a price no greater than their specified limit price), and the two groups then trade within some specified market mechanism. Each trader's individual limit price is private (i.e., is not known by any other trader). Each buyer is encouraged to trade in the market by being instructed to consider the difference between the given limit price and the actual contract price paid for the commodity as pure profit. Furthermore, buyers are told it is better to make no profit and own the commodity than to go without (i.e., they are encouraged to 'trade at the margin'). Similarly, each seller is instructed to consider the difference between the contract price and the given limit price as profit, and to trade at the margin.

Each experiment is run as a sequence of distinct trading periods or 'days'. At the start of each day, all the traders are allowed to make verbal 'shouts' (i.e., quotes) of a price: sellers shout *offers* (e.g., "sell at \$2.50") and buyers shout *bids* (e.g., "buy at \$1.20"). The shouts continue, typically with both groups of traders altering their shout-values (increasing bids and decreasing offers) in an attempt at securing a deal. At any time, a buyer is free to accept a seller's offer or a seller is free to accept a buyer's bid. When this happens, the buyer and seller are considered to have entered into a binding contract to make the deal: for both traders, the number of units they are entitled to trade in is reduced by one, and when a trader's entitlement reaches zero that trader drops out of the market for the remainder of that trading day. This process continues until the shouts of the traders no longer lead to contracts being made, or when some predetermined time-limit (typically five or ten minutes) is reached, at which point the day ends. If there are more days to run, the entitlements of all traders are then restored to their start-of-day values and the market reopens for another day of business.

The distribution of limit prices determines the supply and demand curves for the experiment, and their intersection indicates the theoretical equilibrium price and quantity. In a typical experiment, trading in the first day is characterised by early transactions taking place at prices that differ significantly from the equilibrium value: as the day progresses, the transaction prices approach the equilibrium value. On subsequent days, the transaction prices are initially nearer equilibrium, and approach equilibrium faster. In most of the experiments described by Smith (1962), only the prices of agreed transactions were recorded. However, it can be informative to record all the prices shouted by the traders, including offers and bids that are not accepted (e.g., Smith, 1962, Chart 10). Qualitative illustration of such data for an idealised experiment is shown in Figure 2.

To better characterize the convergence of transaction prices, (referred to as 'exchange prices' in Smith's paper), Smith introduced a "coefficient of convergence", α, that is computed at the end of each day's trading in the market. This metric is defined as $\alpha = 100\sigma_0/P_0$, where "$\ldots\sigma_0$ is the standard deviation of exchange prices around the equilibrium price rather than the mean exchange price." (Smith, 1992, p.13).[2] In most of Smith's experiments, α tends to decline from one trading day to the next. Smith also monitored the *allocative efficiency* of the market. This is defined as the total profit earned by all the traders divided by the maximum possible total profit that could have been earned by all the traders, expressed as a percentage. Typically, after one or two trading periods, human traders achieve allocative efficiency very close to 100%.

In the first eight experiments reported by Smith (1962), each trader is allowed to buy or sell only one unit, although in later experiments this constraint was relaxed. Smith also experimented with changing the supply and demand during the experiment (i.e., after a few trading days, before the start of the next day's trading, a new set of limit prices were given to the subjects), and with having the buyers remain silent while only the sellers were allowed to shout

[2]Formally, for a set of n transaction prices denoted by P_i: $i \in \{1, \ldots, n\}$; $\sigma_0 = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i - P_0)^2}$.

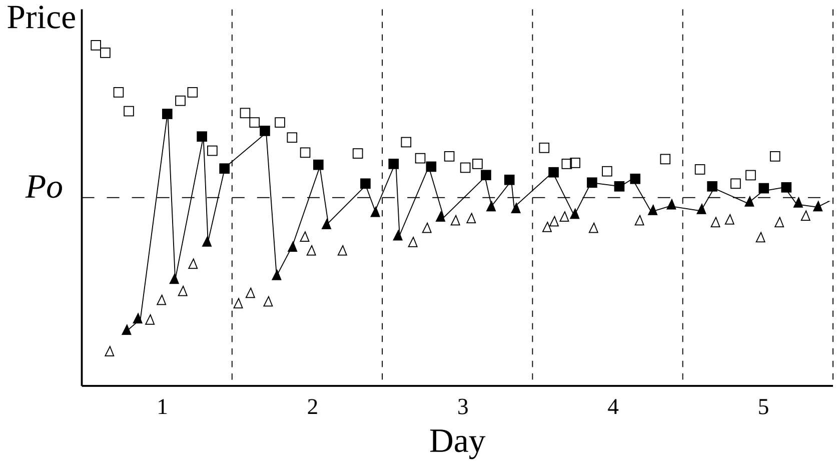

Figure 2: Time series of shout prices for bids and offers, both accepted and rejected. This synthetic data shows an idealization of the market process in one 5-day experiment. Shout prices for bids are shown as triangles, while offers are shown as squares. Open symbols are shouts that were ignored, while filled symbols are shouts that were accepted. The horizontal dashed line indicates the theoretical equilibrium price P_0, and the vertical dashed lines indicate the end of trading periods or 'days'. For clarity, a line joins the sequence of accepted bids and offers: as the experiment progresses, the transaction prices approach equilibrium, and on successive days there is less variance from, and faster approach to, the value of P_0.

offers: an experiment discussed in more detail in Section 6.4. Smith's results qualitatively indicated that the relationship of the supply and demand curves had an impact on the way in which transaction prices approached equilibrium: whether equilibrium was approached from above or below the predicted value, and whether the value was actually reached or the traders stabilised at an off-equilibrium value. Discussing these results in a subsequent publication, Smith states:

> "What have I shown? I have shown that with remarkably little learning, strict privacy, and a modest number [of subjects], inexperienced traders converge rapidly to a competitive equilibrium under the double auction mechanism. The market works under much weaker conditions than had traditionally been thought to be necessary. You didn't have to have large numbers. Economic agents do not have to have perfect knowledge of supply and demand. You do not need price-taking behavior – everyone in the double oral auction is as much a price maker as a price taker." (Smith, 1992, p.157).

Smith's (1962) results were some of the first to demonstrate that markets consisting of small numbers of traders could still exhibit equilibration to values predictable from classical microeconomic theory. To appreciate why this is significant, it is necessary to consider the underlying supply and demand curves in more detail.

Consider a simple example where there are five sellers, denoted by the letters A to E, and five buyers, F to J, each with an entitlement to trade one unit. If the limit prices for the sellers are, say, A=\$1.50, B=\$2.00, C=\$2.50, D=\$3.00, and E=\$3.50 then there is an upward-sloping demand curve: at prices less than \$1.50, no sellers are able to sell; at prices between \$1.50 and \$2.00, only A is able to sell; between \$2.00 and \$2.50, both A and B are prepared to sell, so the quantity supplied is two; the increase in quantity supplied continues at higher prices, until a quantity of five is reached at prices above \$3.50, where all the suppliers are able to sell. Similarly, assigning buyer limit prices of F=\$1.50, G=\$2.00, H=\$2.50, I=\$3.00, and J=\$3.50, gives a downward-sloping demand curve: at prices less than \$1.50, all five buyers are able to trade, and as higher prices are considered so the quantity demanded falls, until the value of \$3.50 is reached, above which no buyers are able to trade. The corresponding supply and demand curves are illustrated in Figure 3. From the figure, it is clear that the equilibrium price is \$2.50, and the clearing quantity is three. If traders were not actively encouraged to enter into marginal deals, the equilibrium price would still be \$2.50 but the clearing quantity would be either two or three: the horizontal overlap of supply and demand creates what Davis and Holt (1993, p.131) refer to as a 'volume tunnel'. Entitlements to buy or sell that determines the supply and demand curves to the left of the equilibrium quantity Q_0 are referred to as *intra-marginal* (or infra-marginal) units, while those determining the shape of the curves to the right of Q_0 are *extra-marginal*. Thus the entitlements of traders A, B, J, and I are intra-marginal, while D, E, F, and G are extra-marginal; C and H are simply *marginal*.

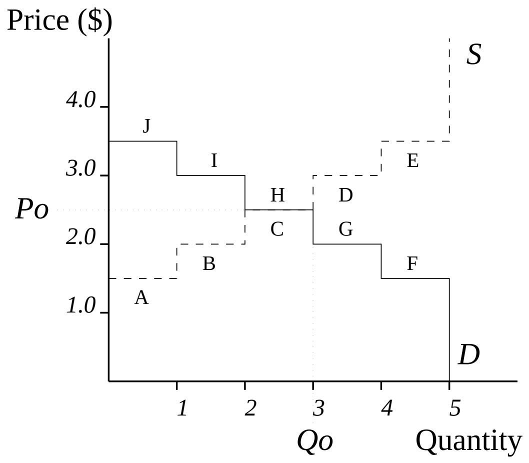

Figure 3: Supply and demand curves for ten traders. The supply curve is shown as a dashed line for extra clarity. There are five sellers (A to E), each with one unit to sell, and five buyers (F to J), each with the right to buy one unit. The step-changes in the quantities supplied and demanded are dependent on the limit prices of the individual traders, as indicated by the labels A to J. The intersection gives equilibrium values P_0=\$2.50 and Q_0=3.

As is evident in Figure 3, the supply and demand curves are stepped. This is because the commodity is dealt in indivisible discrete units, and there are only a small number of units available in the market. Thus, supply and demand in this simple market differs appreciably from the smoothly sloping curves of an idealised market, as was illustrated in Figure 1. The idealised market is based on conditions where the step-changes involved can be treated as infinitesimally small. Most classical theories of price determination and equilibration assume or require a large number of traders: if an individual trader drops out of the market, the supply and demand curves remain essentially the same. But in markets with few participants, this is not the case. Consider the simple market illustrated in Figure 3: if the first trade in the market is between seller C and buyer I (at a price, say, of \$2.65: giving profits of \$0.15 for C and \$0.35 for I) then these two traders drop out of the market, but the resultant supply and demand curves, illustrated in Figure 4, are significantly different. Now the equilibrium quantity is reduced to two, and the equilibrium price is indicated at \$2.25, although technically it is no longer a scalar value: rather there is now a bounded range of possible equilibrium prices, from \$2.00 to \$2.50:

what Davis and Holt (1993, p.131) refer to as a 'price tunnel'; any price within this range would be an equilibrium value.

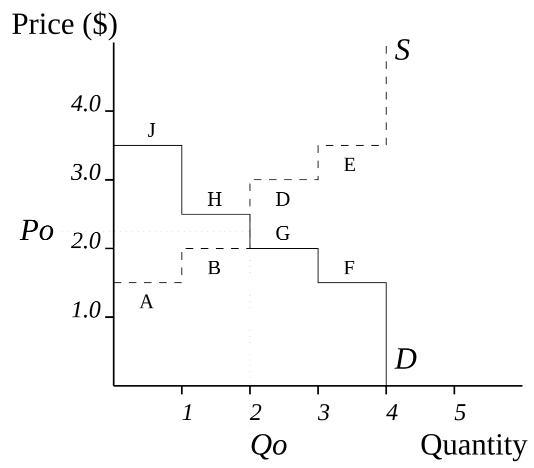

Figure 4: Supply and demand curves for eight traders. This is the market illustrated in Figure 3, after traders C and I have agreed a deal and left the market. The equilibrium price and quantity have altered.

Matters are further complicated when the difference between the *underlying* and *apparent* supply and demand curves is considered. Each active trader in the market will be trying to make a profit, so buyers will be shouting prices lower than their individual limit prices, and sellers will be shouting prices higher than their limit prices. Because the limit price of each trader is private (i.e., known only to that trader), the prices shouted by the traders give only an indication of the *underlying* supply and demand determined by the limit prices: the *apparent* supply and demand, based on the actual prices shouted, may be significantly different. Thus, Figures 3 and 4 show underlying supply and demand before and after the trade between C and I, but an observer of (or participant in) the market does not have access to this information: these underlying schedules can only be guessed at by the traders, and the only information they have available is the shout-prices observed (i.e., "heard") in the market.

Smith (1992, pp.809–810) refers to the underlying supply and demand as the *market* supply and demand, and to the apparent supply and demand as the *seller's offer array* and *buyer's bid array*: referred to collectively as the bid-and-offer arrays (Davis & Holt, 1993, p.300).

To illustrate the difference, assume that each trader has a 'profit level' that he or she aims to achieve in selling or buying, and express this as a percentage of the trader's limit price. So a seller with a limit price of \$2.00 and a profit level of 10% will shout a price of \$2.20, while a buyer with the same limit price and profit level will shout a price of \$1.80. Now take the market of Figure 3 and assign each trader a randomly chosen profit level in the range 0% to 50%. For example: A=35%, B=5%, C=15%, D=35%, E=10%, F=20%, G=30%, H=5%, I=25%, and J=10%. From these percentages, we can calculate the prices each trader would shout: A=\$2.03, B=\$2.10, C=\$2.88, D=\$4.05, E=\$3.85, F=\$1.20, G=\$1.40, H=\$2.38, I=\$2.25, and J=\$3.15. Now if we assume that, at the start of the day, all traders shout their prices simultaneously (unlikely, but equivalent to a single-round sealed-bid auction), then the prices heard in the market give the apparent supply and demand curves illustrated in Figure 5. As can be seen, the apparent supply and demand differ significantly from the underlying curves illustrated in Figure 3. The equilibrium price and quantity are different, and the ranking of the trader's prices has altered.

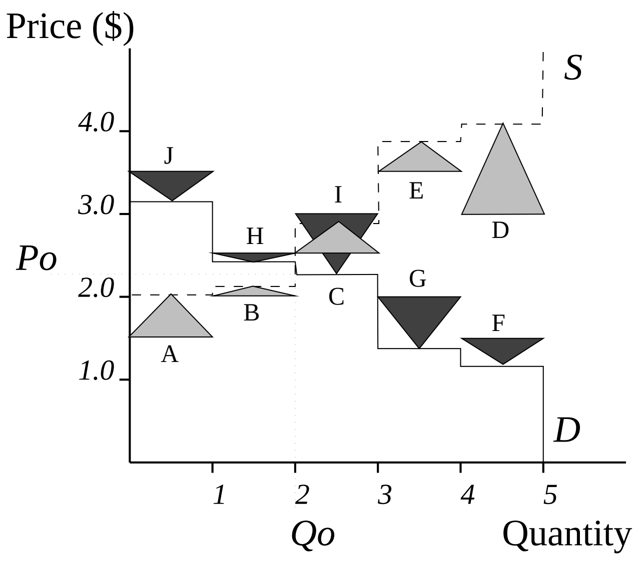

Figure 5: Bid-offer array for the ten-trader market illustrated in Figure 3, given random profit levels between zero and fifty percent. Each buyer's limit and shout prices are illustrated as dark inverted triangles, while each seller's limit and shout prices are illustrated by light upright triangles: the base of each triangle indicates the trader's limit price, while the apex indicates the trader's shout-price. The array of bid-prices gives an apparent demand curve D, and the array of offer-prices gives an apparent supply curve S. The apparent supply and demand curves differ significantly from the underlying supply and demand shown in Figure 3: see text for discussion.

Finally, it should be noted that the apparent supply and demand schedules are dynamic, and can alter rapidly. Because no trader has full knowledge of the underlying supply and demand, traders might base their profit levels on an initial guess that is then refined on the basis of hearing the prices shouted by the competition (other members of the trader's group) and opposition (traders in the other group, or 'contraside' (Bollerslev & Domowitz, 1992, p.226)), and on the basis of which shouts lead to deals and which are ignored. In an open-outcry continuous double auction, such information arrives in a continuous asynchronous stream. Moreover, the underlying supply and demand dynamically shift as traders make deals and leave the market (discussed further in Section 6.4).

Despite this, the humans in Smith's experiments rapidly approach the competitive equilibrium predicted from classical rational-expectations theory. Figures 6 to 9 show some of Smith's (1962) results. In all four figures, the supply and demand curves are shown on the left, and the time series of transaction prices over a number of trading days is shown on the right. In Figure 6, there are 11 buyers and 11 sellers, each with the right to buy or sell one unit: $P_0 = \$2.00$; $Q_0 = 6$. The shaded area in Figure 6 indicates the available 'surplus' or 'rent' in the market: this is divided into two regions by the horizontal line at P_0, and Smith hypothesised that the ratio of the areas of these two regions affected the nature of the approach of transaction prices to the theoretical equilibrium price. As is clear on the right of Figure 6, transaction prices converge toward equilibrium over the five trading days, and the number of transactions per day varies from five to seven.

In Figure 7, the supply curve is 'perfectly elastic' (i.e., flat). As can be seen, the transaction prices approach equilibrium from above, but level out at prices approximately \$0.20 above equilibrium. In the original experiment, Smith attempted to 'shock the market down to equilibrium' by decreasing demand at the end of Day 4 and continuing the experiment for another three days, but this had no discernible effect on transaction prices.

In Figure 8, there is excess demand (12 sellers but 17 buyers): transaction prices converge to equilibrium very slowly, and from below: when equilibrium is reached, there is evidence of some 'overshoot'. In the original experiment, Smith reduced demand after Day 4 and continued the experiment for two further days: the market quickly converged (from above) on the new equilibrium, and did not overshoot it.

In Figure 9, supply and demand are similar to Figure 6 but only sellers were allowed to shout offer-prices: buyers played a passive role, making no bids, and staying silent until they accept an offer. This was intended to model a retail market (rather than a continuous double auction): transactions start in Day 1 at above-equilibrium values, but over the course of the first day they fall below the equilibrium price, and remain low for the next three days. In the original experiment, Smith allowed the buyers to announce bids after Day 4, continuing for two more days: the change in market organisation was reflected by transaction prices moving back to values at and above the equilibrium price. This particular experiment is discussed in more depth in Section 6.4.

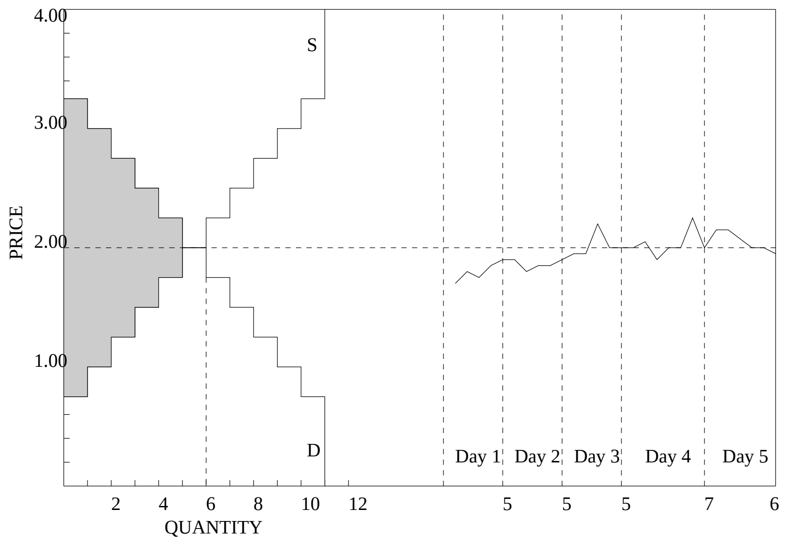

Figure 6: Redrawn from Smith's (1962) Chart 1: see text for discussion.

Human beings are notoriously smart creatures: the question of just how much intelligence is required of an agent to achieve human-level performance is an intriguing one. This question was addressed by Gode and Sunder, whose work is summarised in the next section. Gode and Sunder's results indicate that no intelligence is required, but this claim is called into doubt in Section 5. Following that, Section 6 demonstrates that, although zero intelligence is not enough, simple mechanisms similar to those just described (i.e., each trader shouting a price determined by a combination of that trader's limit price and profit level, with the profit level being adjusted on the basis of other shouts in the market) can give performance very similar to that of groups of human traders.

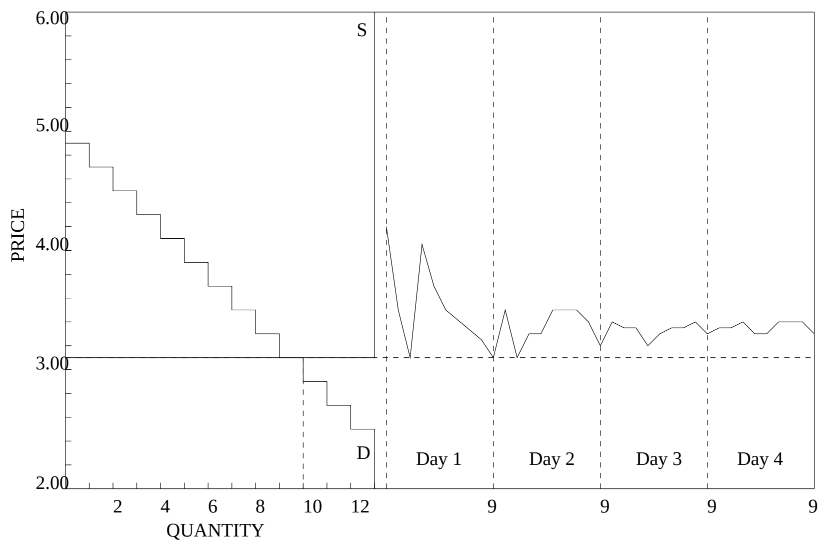

Figure 7: Adapted from from Smith's (1962) Chart 4. See text for discussion.

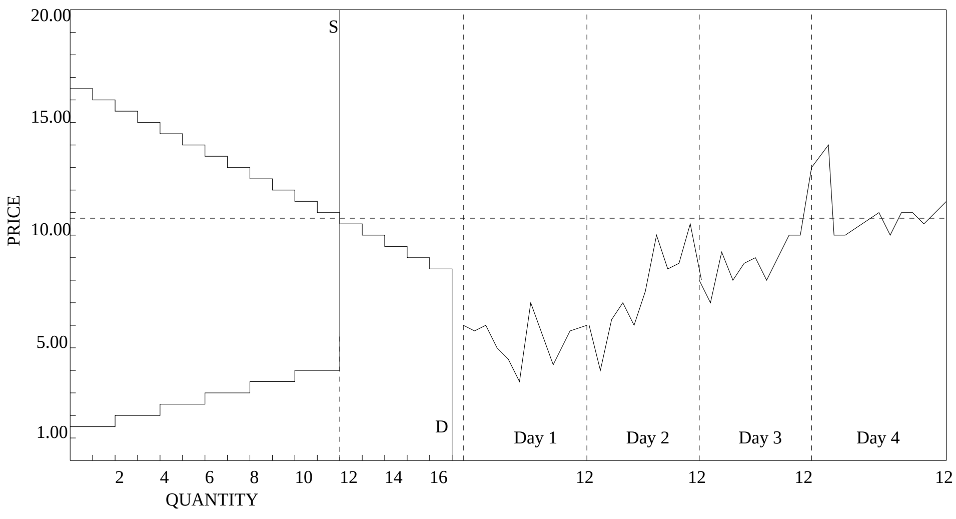

Figure 8: Adapted from Smith's (1962) Chart 6. See text for discussion.

4.2 Gode and Sunder's Zero Intelligence Traders

Gode and Sunder (1993) describe a set of experiments similar in style to Smith's, but which use "zero intelligence" (ZI) programs that submit random bids and offers to replace human traders in electronic double-auction markets. They explore the performance of both 'unconstrained' and 'constrained' zero-intelligence traders, which they refer to using the abbreviations ZI-U and ZI-C, and compare the results of these traders to results from human traders operating in (almost) identical experimental conditions.

As with Smith's work, each trader is given an entitlement to buy or sell a number of units,

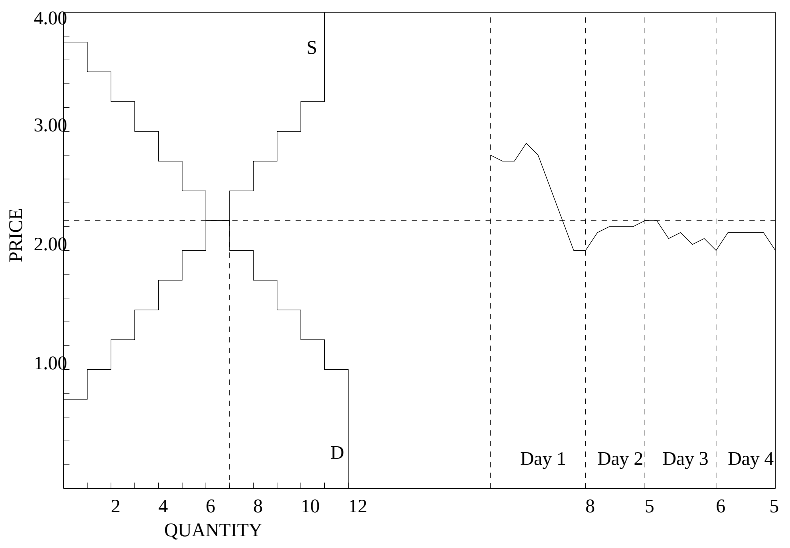

Figure 9: Adapted from Smith's (1962) Chart 8. See text for discussion.

each with a particular limit price. Bids and offers were limited to the range 1 to 200 units of arbitrary currency (which I'll write as \$0.01 to \$2.00, for consistency with the previous discussions). For ZI-U traders, the shout-price for a unit of commodity is unconstrained across this range: there is a 1/200 probability that the value of a random shout is v. This means that that the ZI-U traders can enter into loss-making deals: a seller of a unit of commodity could shout a price less than the given limit price, or a buyer could shout a price greater than the limit; in either case, if the shout is accepted by another trader, the agent makes a loss. In contrast, ZI-C traders are subject to a 'budget constraint' that prevents them from engaging in loss-making deals: a ZI-C seller can only make offers in the range between the limit price and the \$2.00 maximum; and ZI-C buyers can only make bids in the range from the \$0.01 minimum up to their limit price. The shout prices are generated from a uniform distribution across the range for each ZI-C trader.

The experiments with both types of ZI traders were conducted using minor simplifications of the NYSE continuous double auction, with a transaction cancelling any unaccepted bids and offers. The traders dealt in lot-sizes of a single unit of commodity. To accommodate the lack of intelligence of the traders, a deal was made whenever a bid and offer crossed: whenever a buyer made a bid higher than the current lowest offer, or whenever a seller made an offer lower than the current highest bid. In both cases, the transaction price is the earlier of the two shouts.

Gode and Sunder (1993) report on ZI traders with the right to buy or sell multiple units of commodity during the course of the experiment, with the lot-size for each offer, bid, and deal fixed at one unit. They state (1993, p.122) that experiments with a single unit per trader (reported in (Gode & Sunder, 1992)) gave similar results, although the single-unit-per-trader studies were not concerned with the behavior of prices (Gode & Sunder, 1992, p.202). The units given to an individual seller may have different limit prices, as may the purchase rights given to an individual buyer. In one experiment, several traders were given limit prices for all their units that were extra-marginal (beyond the equilibrium point and therefore difficult to trade) while the other traders' assignments were all intra-marginal units. In the other experiments, each trader had a spread of limit prices across the range of supply or demand. To avoid requiring the traders to make a decision about which of their stock of units they should sell next, or which of their rights to buy they should exercise next, the rights to buy or sell were assigned to each agent in a pre-specified order of execution.

Differences in performance between the ZI-U and ZI-C traders, and between the ZI-C and human traders, could indicate the different extents to which overall market behavior is dependent on human intelligence or market structure:

> "The difference between the performance of the human markets and that of the ZI-C markets is attributable to systematic characteristics of human traders. If ZI-C traders are considered to have zero rationality, this difference in performance would be a measure of the contribution of human rationality to market performance. On the other hand, the difference between the performance of markets that do impose a budget constraint on ZI traders and the performance of those that do not is attributable to the market discipline. Traders have no intelligence in either the ZI-U or ZI-C market: the ZI-C market prevents the traders from engaging in transactions that they cannot settle. Consequently, we can attribute the differences in market outcomes to the discipline imposed by the double auction on traders" (Gode & Sunder, 1993, p.123).

Results from five experiments are reported in the 1993 paper. Each experiment is repeated three times: once with ZI-U traders (six buyers and six sellers), once with ZI-C traders (six buyers and six sellers), and once with human subjects (six or seven buyers and six or seven sellers, all of whom were business graduate students, given an incentive to do well by having their profits included in the grading scheme for their course). Like the ZI-C traders, the humans were subject to the budget constraint: they could (or should) not enter into loss-making deals. Although the supply and demand schedules varied between experiments, in any one experiment the ZI-U and ZI-C traders had identical supply and demand schedules, and the humans schedules differed only slightly, when there was an extra trader or buyer. Nevertheless the human schedules were very similar to those of the ZI traders, and had the same equilibrium price and quantity.

Figure 10 gives an indication of the qualitative differences between the price histories of the three types of trader: in all five experiments, the same qualitative differences emerged. Prices in the ZI-U markets exhibited "...little systematic pattern and no tendency to converge toward any specific level" (Gode & Sunder, 1993, p.126). Price histories in the human markets were similar to those in Smith's early experiments: the transaction prices soon settle to stable values close to the theoretical equilibrium price, after some initial learning and adjustment. As Gode and Sunder (1993, pp.127–128) note, such price series are "...the result of subjecting profit-motivated, intelligent human traders to market discipline", and the main question addressed in their article is to what degree the difference between the market activity of the human and ZI-U traders is due to the intelligence of the traders, and to what degree the difference is due to the imposition of the budget constraint. It is this issue that the data from ZI-C traders helps resolve: they have the same budget constraint as the humans, but none of the intelligence or ability to learn from experience.

Gode and Sunder (1993, p.129) point out three notable features of the ZI-C transaction-price time-series. First, as would be expected from traders that have no memory or adaptation mechanisms, there is no evidence of learning from day to day. Second, the volatility of the ZI-C prices is intermediate between the highly volatile ZI-U prices and the stable human prices: the presence of a budget constraint is sufficient to shift the ZI market behavior towards that of humans. Third, despite being more volatile than the human price series, the ZI-C prices

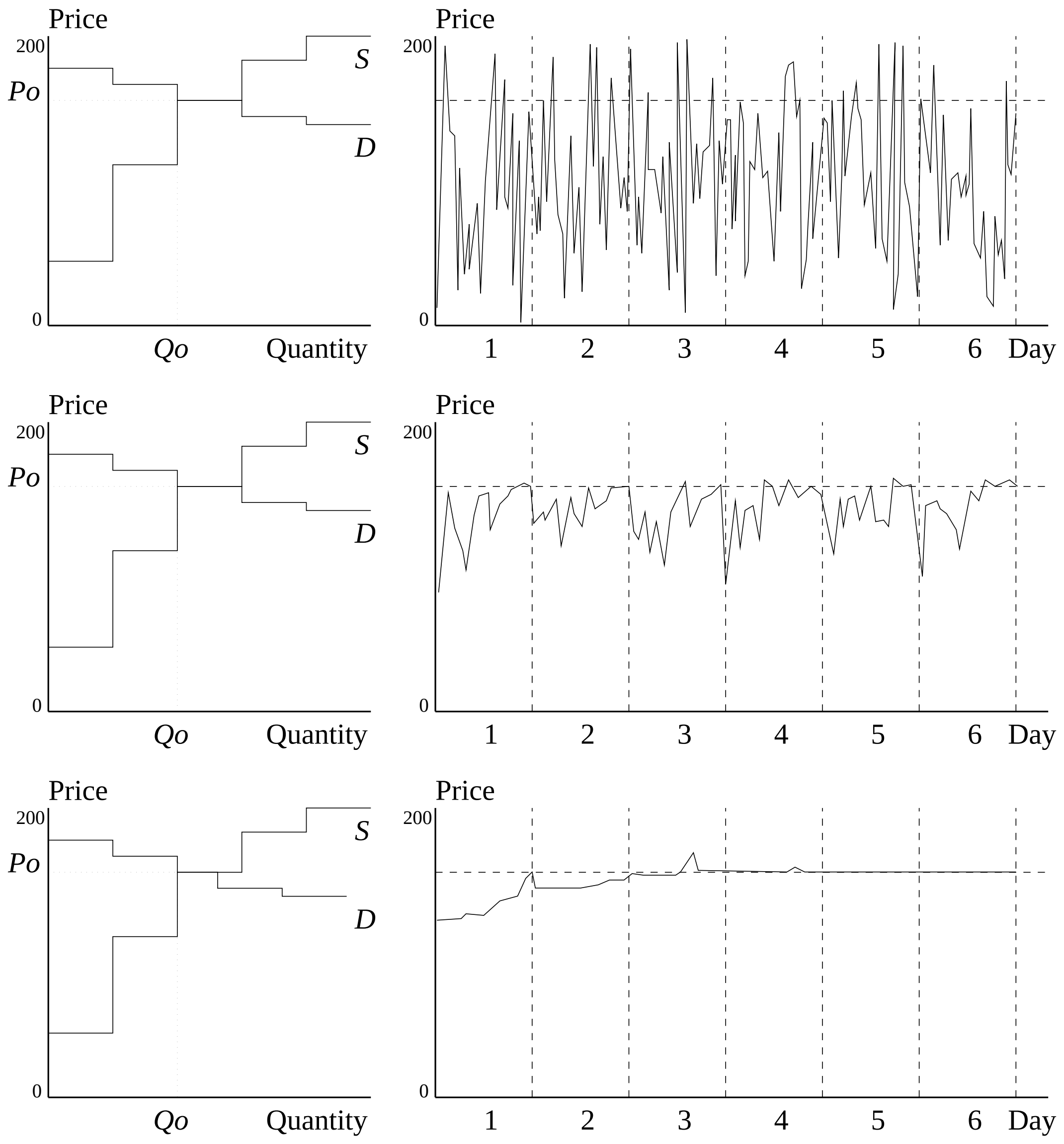

Figure 10: Typical results from one of Gode and Sunder's experiments (redrawn from Gode and Sunder (1993, Fig.4, p.127)). Top: results from ZI-U traders. Middle: results from ZI-C traders. Bottom: results from human traders. The figure to the left shows the supply and demand schedules for the experiment, the figure to the right shows the time series of transaction prices.

converge slowly to equilibrium during each day's trading (this is clear in the middle transaction-price time-series in Figure 10). Gode and Sunder confirm that this convergence is consistent by regression analysis of Smith's α convergence measure, averaged across the six days of a market, versus transaction sequence number.

Gode and Sunder explain this convergence to equilibrium by reference to the progressive narrowing of the feasible range of transaction prices as more units are traded:

> The left end of the market demand function represent units with higher redemption values [i.e., limit prices]. Expected values of the bids generated for these units by ZI-C traders are also higher. Therefore, these units are likely to be traded earlier

than units further down the market demand function. As the higher-value units are traded, the upper end of the support of ZI-C bids shifts down. Similarly, as the lower-cost units are sold earlier in a period, the lower end of the support of ZI-C offers moves up. (Gode & Sunder, 1993, p.129).

In addition to showing human-like tendency to equilibrium, ZI-C traders also exhibit surprisingly high levels for Smith's measure of allocative efficiency. In all five experiments, the mean efficiency of human traders was over 90%, and in four of them it was over 99%. In contrast, the ZI-U traders recorded mean efficiencies of 90% in two experiments, and 77%, 49%, and 86% in the others. But the ZI-C traders scored over 99% in three experiments, and over 97% in the other two: the average efficiency for the humans was 97.9%, while for the ZI-C's it was 98.7%. Gode and Sunder (1993, pp.131–133) did not perform any statistical tests, but the difference of 0.8% between the human and ZI-C traders appears unlikely to be significant. Thus, the main message of Gode and Sunder's paper is that allocative efficiency appears to be almost entirely a product of market structure: prior to these experiments, it seemed fair to assume that the high efficiency of human markets is a consequence of human cognitive prowess; in light of Gode and Sunder's results, such assumptions are clearly highly doubtful.

Gode and Sunder close their paper with brief discussion of measurements of *profit dispersion*. This is defined (1993, p.133) as the cross-sectional root mean squared difference between the actual profits and the equilibrium profits of individual traders. The equilibrium profit of a trader is the profit the trader would realise if all units are traded at the equilibrium price. Formally, if a_i is the actual profit earned by trader i, and p_i is the theoretical equilibrium profit for that trader, then for a group of n traders the profit dispersion is given by $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - p_i)^2}$.

Measures of profit dispersion were highest for the ZI-U traders, and lowest for the human traders. The values for ZI-C traders were closer to those of the humans than the ZI-U traders, but were generally greater in magnitude (again, no statistical significance tests were performed). In two of the five experiments, the human data showed a definite decline in dispersion over a number of trading days: "Without memory or learning, the ZI markets exhibit no such trend. These results suggest that, in contrast to aggregate efficiency, distributional aspects of market performance may be sensitive to human motivation and learning." (Gode & Sunder, 1993, p.134).

While Gode and Sunder's work is elegant and yields impressive results, there are some questionable aspects. For instance, the maximum and minimum values for shout-prices have to be specified in advance, which implies that *a priori* information is employed. Furthermore, it is not obvious how ZI-C traders could be used in situations such as Smith's experiment where only sellers shout (at the very least, some extension of Gode and Sunder's methods would have to be specified). However, these are relatively minor superficial criticisms. A much more fundamental critique of Gode and Sunder's work is presented in the next section.

5 Critique: Zero is not enough

Gode and Sunder's central argument, that the structure of a double auction market is largely responsible for achieving high levels of allocative efficiency, regardless of the intelligence, motivation, or learning of the agents in the market, is not in doubt. However, in this section, the equilibrating tendencies of the ZI-C traders is questioned. Gode and Sunder state (1993, p.131):

> "...the convergence of transaction price [to the theoretical equilibrium price] in ZI-C markets is a consequence of the market discipline; trader's attempts to maximize their profits, or even their ability to remember or learn about events of the market, are not necessary for such convergence."

This statement is proven below to be incorrect. In Section 5.1 the statistics of the ZI-C markets are discussed qualitatively, noting that the average transaction prices observed in the market are determined by the intersection of the probability density functions for the buyers' and sellers' random bids and offers. The mean or expected value of the transaction-price distribution is shown qualitatively to be close to the equilibrium price only in situations where the magnitude of the gradient of linear supply and demand curves is roughly equal. Then, in Section 5.2, analytic results are presented that demonstrate that the expected value is equal to the equilibrium price only in certain special cases, and that the expected value differs significantly in other situations. To reinforce this result, empirical results from simulation studies are presented in Section 5.3, which are discussed further in Section 5.4.

5.1 Qualitative Discussion

Parameters relevant to the discussion that follows are shown in Figure 11.

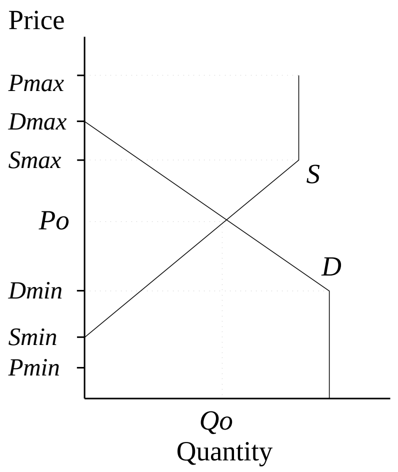

Figure 11: Parameters for qualitative discussion. Offers and bids are both subject to a maximum possible price $P_{\max}$ and a minimum possible price $P_{\min}$. The Supply Curve S slopes upwards from $S_{\min}$ to $S_{\max}$ and the Demand Curve D slopes downward from $D_{\max}$ to $D_{\min}$. The Supply and demand curves intersect at a point indicating the Equilibrium Price P_0 and the Equilibrium Quantity Q_0. In this graph $D_{\min} > S_{\min}$ and $D_{\max} > S_{\max}$ but these conditions are not mandatory.

Because the ZI traders randomly generate bid and offer prices within given upper and lower limits, and assuming (without loss of generality) that the prices vary continuously between those limits, it is possible to construct probability density functions (PDFs) for the prices of bids and offers shouted in the market.

In the case of ZI-U traders, the construction of the PDF is trivial: as noted by Gode and Sunder (1993, p.121), for prices in the range 1 to 200, the probability that a randomly generated bid or offer is nearest some integer value i is $1/200$ ($= 0.005$) for all $i = 1, 2, \ldots, 200$. Thus a graph of the ZI-U PDF shows a uniform distribution of probabilities, as illustrated in Figure 12.

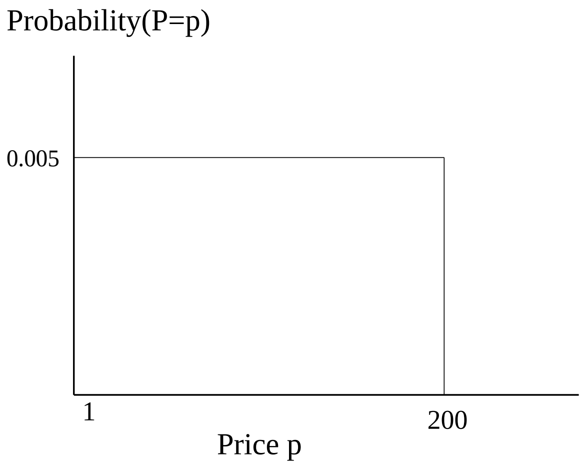

Figure 12: Probability density function (PDF) for prices 'shouted' by ZI-U traders: this PDF applies both to the sellers's offer-prices and the buyers' bid-prices.

However, the ZI-C traders have slightly more complex PDFs. Consider the case for a ZI-C seller: it can only generate an offer price between its allocated limit price, and the predetermined system-wide maximum price P_{max} for bids or offers ($P_{\text{max}} = 200$ in Gode and Sunder's experiments), and the seller's individual PDF is uniform over that range. Thus, for the market as a whole, the sellers' supply curve acts as a lower bound on the offer prices generated at any given quantity (shown in Figure 13), and so the PDF for offer prices in the market rises from zero to some threshold value, at which it might plateau, before falling back sharply to zero at P_{max} (Figure 14). This qualitative description can be justified as follows: there is no probability of an offer price below the lowest seller's cost (S_{min}); as prices higher than S_{min} are considered, more offers are likely because there are more sellers able to bid; once prices greater than the maximum seller's cost (S_{max}) are considered, increasing prices are not increasingly likely as all sellers are able to make offers in that price range (hence the potential for plateau); and the probability of an offer above the system maximum P_{max} is zero.

Similarly, the range of possible prices for bids from ZI-C buyers is bounded from above by the demand curve (formed by the buyers' limit prices) and from below by the system-wide minimum price P_{min} for bids or offers (i.e. $P_{\text{min}} = 1$ in Gode and Sunder's experiments): see Figure 15. Thus the PDF for bid prices in the market is zero at prices below P_{min}, constant for prices between P_{min} and the minimum buyer limit price D_{min}, then falls gradually to reach zero at the highest buyer limit price D_{max}: see Figure 16.

Having established qualitative PDFs for the *offers* generated by the sellers and the *bids* generated by the buyers, we can now consider the PDF for *transaction* prices: recall that a transaction is made when a buyer's bid is accepted because it is higher than the current best (lowest) offer, or when a seller's offer is accepted because it is lower than the current best (highest) bid. But the current best bid and offer must be valid, i.e. must come from one of

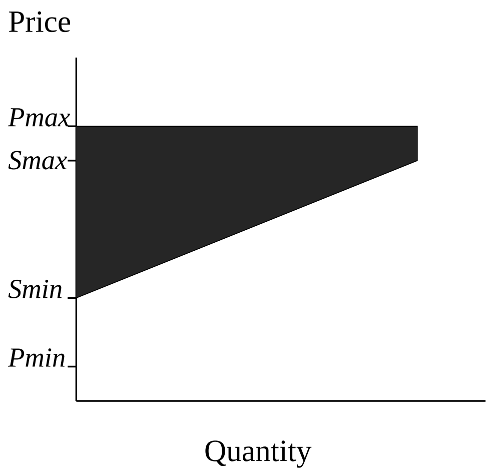

Figure 13: The supply curve provides a lower bound on the range of possible offer prices from ZI-C sellers at any given quantity; the system maximum $P_{\max}$ provides an upper bound.

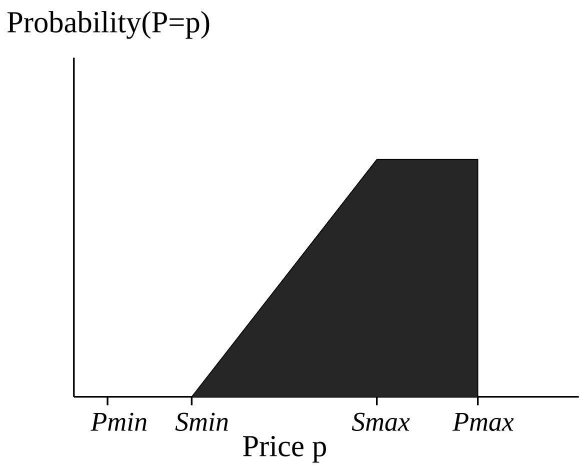

Figure 14: Qualitative probability density function (PDF) for ZI-C sellers: see text for discussion.

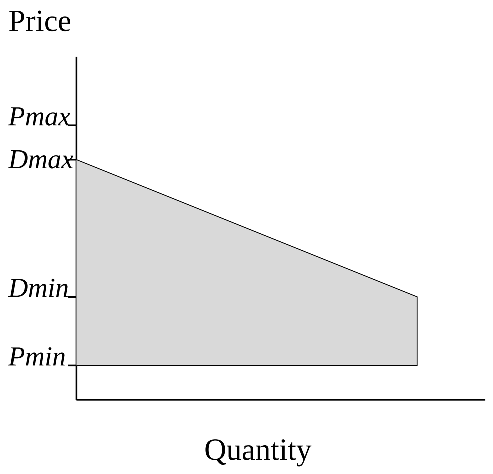

Figure 15: The demand curve provides an upper bound on the range of possible bid prices from ZI-C buyers at any given quantity; the system minimum $P_{\min}$ provides a lower bound.

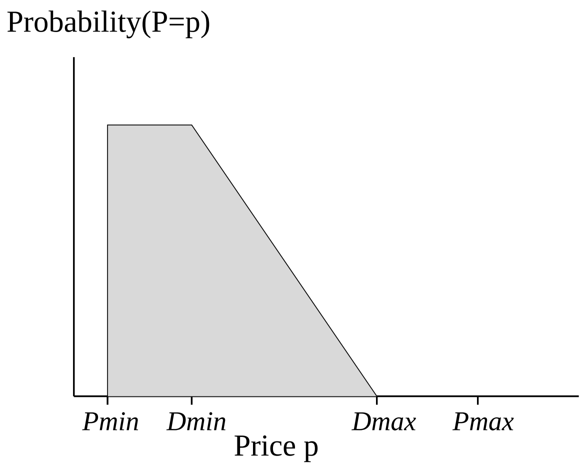

Figure 16: Qualitative probability density function (PDF) for ZI-C buyers: see text for discussion.

the PDFs in Figures 14 and 16. Thus, the PDF for transaction prices will be determined by the *intersection* of the PDFs of the offer prices and bid prices: transactions require a valid bid *and* a valid offer; despite the fact that many bids may be made at prices lower than $S_{\min}$, no seller can accept one; and although many offers may be made at prices higher than $D_{\max}$, no buyer can accept one. Figure 17 shows the intersection of the PDFs for offer and bid prices; and the corresponding PDF for transaction prices.

Finally, note that in Figure 17 the peak of the PDF, indicating the most probable transaction price(s), is very close to the Equilibrium Price P_0 (cf. Figure 11). Thus, qualitatively at least, it would appear that near-equilibrium transaction prices are expected because of the shape of the PDF for valid deals: this ZI-C system is structured *a priori* to generate mean transaction prices

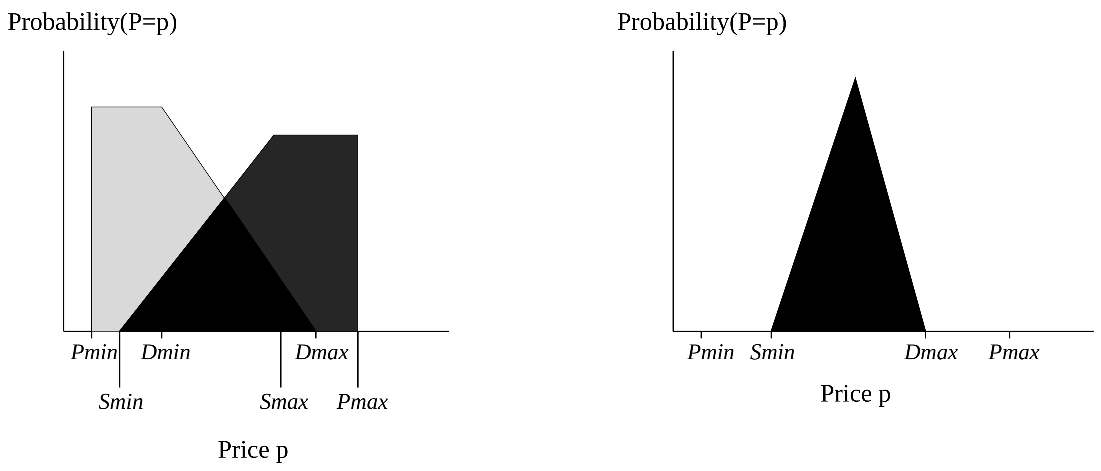

Figure 17: Left: darkest-shaded triangular area shows the intersection of the PDFs for offer and bid prices; this area indicates the PDF for transaction prices, but requires normalisation so that the area of the triangle is unity. Right: corresponding normalised PDF for transaction prices.

close to the theoretical equilibrium price.

If this is true, it is only because the transaction price PDF has a shape closely approximating an isosceles triangle, which is a consequence of the supply curve S and the demand curve D being *symmetric* (i.e., having gradients of opposite sign but approximately equal magnitude). As is proven in the next section, when the linear S and D curves have gradients opposite in sign and *identical* in magnitude, the mean transaction price is *identical* to the equilibrium price: the equilibrating tendency of the ZI-C traders is thus a consequence of the underlying statistics of the system. Furthermore, it is demonstrated below that, in general, the mean transaction price of ZI-C traders differs significantly from the theoretical equilibrium price.

5.2 Analytic Arguments

Let $f(p)$ denote the PDF for transaction prices. If $f(p)$ is known, then the mean or expected value $E(P)$ of the transaction prices can be calculated from the standard formula:

$$E(P) = \int_{-\infty}^{\infty} p \cdot f(p)\, \mathrm{d}p \tag{1}$$

Consider the case where the supply and demand curves are symmetric (i.e., have opposite sign and equal magnitude), as illustrated in Figure 18. The corresponding PDF is shown in Figure 19. Such a market is similar to Smith's (1962) 'Chart 1' (see Figure 6).

The transaction-price PDF can be written as:

$$f_1(p) = \begin{cases} 0 & \text{if } p < S_{\min} \\ m_1 p + c_s & S_{\min} \le p \le P_0 \\ -m_1 p + c_d & P_0 \le p \le D_{\max} \\ 0 & p > D_{\max} \end{cases} \tag{2}$$

Let $k = D_{\max} - P_0$; note that, as the PDF is an isosceles triangle, $k \equiv P_0 - S_{\min}$ and $h_1 k = 1$. Also, from Figure 19, $m_1 = h_1/k = 1/k^2$. Therefore, substituting Equation 2 into Equation 1 gives:

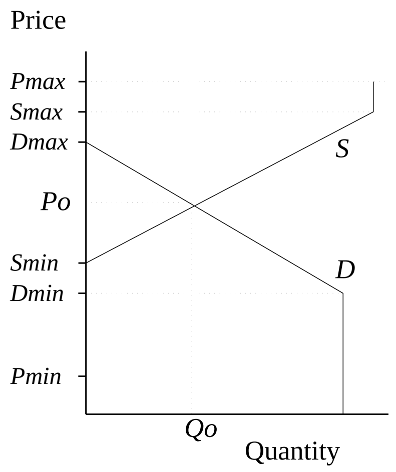

Figure 18: Symmetric supply and demand.

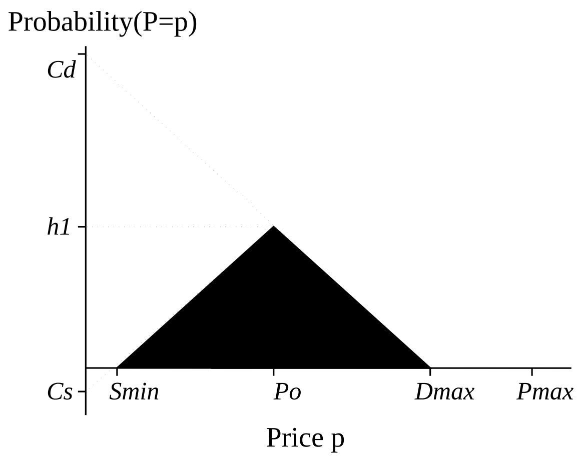

Figure 19: Transaction-price PDF for supply and demand curves of Figure 18: see Equation 2.

$$
\begin{aligned}
E(P) &= \int_{S_{\min}}^{D_{\max}} p f_1(p)\,\mathrm{d}p \\
&= \int_{S_{\min}}^{P_0} p(m_1 p + c_s)\,\mathrm{d}p + \int_{P_0}^{D_{\max}} p(-m_1 p + c_d)\,\mathrm{d}p \\
&= P_0 + \int_{-k}^{0} p(m_1 p + h_1)\,\mathrm{d}p + \int_{0}^{k} p(-m_1 p + h_1)\,\mathrm{d}p \\
&= P_0 + \left[\frac{1}{3}m_1 p^3 + \frac{1}{2}h_1 p^2\right]_{-k}^{0} + \left[-\frac{1}{3}m_1 p^3 + \frac{1}{2}h_1 p^2\right]_{0}^{k} \\
&= P_0 + \frac{1}{3}m_1 k^3 - \frac{1}{2}h_1 k^2 - \frac{1}{3}m_1 k^3 + \frac{1}{2}h_1 k^2 \\
&= P_0 \ \square \qquad (3)
\end{aligned}
$$

Thus, Equation 3 demonstrates that when the supply and demand curves are linear, having opposite sign and equal magnitude, the mean transaction price $E(P)$ is equal to the equilibrium price P_0.

Now consider the case for a PDF where the supply curve is flat, so that $S_{\min} = S_{\max} = P_0$ (see e.g. 'Chart 4' in Smith (1962), shown in Figure 7). Such a supply curve is illustrated in Figure 20, with the corresponding transaction-price PDF shown in Figure 21.

The PDF $f_2(p)$ is given by:

$$
f_2(p) = \begin{cases} 0 & \text{if } p < P_0 \\ m_2 p + c_2 & P_0 \le p \le D_{\max} \\ 0 & p > D_{\max} \end{cases} \qquad (4)
$$

For $m_2 = -h_2/j$ where $j = D_{\max} - P_0$, and $c_2 = 2P_m/j^2$. Note also that because $f_2(p)$ is a PDF and a right-triangle, $h_2 j/2 = 1$, so $h_2 = 2/j$ and hence $m_2 = -2/j^2$. Substituting Equation 4 into Equation 1 gives:

$$
E(P) = \int_{P_0}^{D_{\max}} p f_2(p)\,\mathrm{d}p
$$

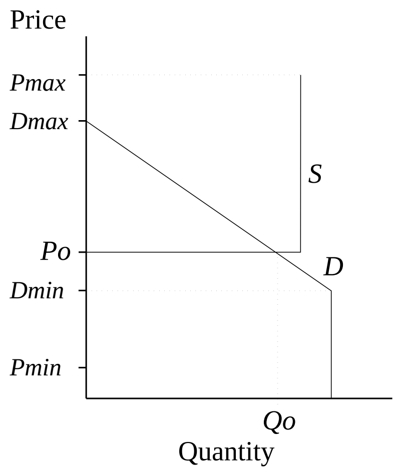

Figure 20: Flat supply curve.

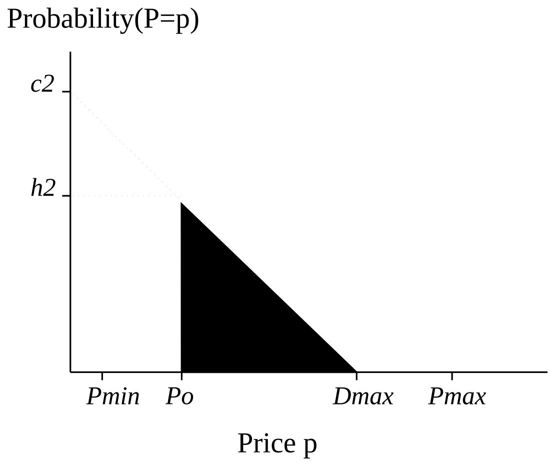

Figure 21: PDF for flat supply.

$$
\begin{aligned}
&= P_0 + \int_0^j m_2 p^2 \, \mathrm{d}p + \int_0^j h_2 p \, \mathrm{d}p \\
&= P_0 + \left[\frac{1}{3} m_2 p^3\right]_0^j + \left[\frac{1}{2} h_2 p^2\right]_0^j \\
&= P_0 + \frac{1}{3} m_2 j^3 + \frac{1}{2} h_2 j^2 \\
&= P_0 + \frac{1}{3} \cdot \frac{-2}{j^2} j^3 + \frac{1}{2} \cdot \frac{2}{j} j^2 \\
&= P_0 + \frac{j}{3} \\
&= P_0 + \frac{1}{3}(D_{\max} - P_0) \quad \square \qquad (5)
\end{aligned}
$$

So Equation 5 indicates that, when all the sellers have the same limit price, the expected transaction price of ZI-C traders will differ from the equilibrium price P_0 by an amount equal to one third of the difference between P_0 and the maximum buyer price, $D_{\max}$.

Finally, consider the case of "box design" schedules (Davis & Holt, 1993, p.141), where both the supply curve and the demand curve are flat, so that $S_{\min} = S_{\max}$ and $D_{\min} = D_{\max}$. Such markets are similar to Smith's (1962) 'Chart 6' (see Figure 8). If demand equals supply, then there is an indeterminate 'price tunnel' and the equilibrium price cannot be accurately predicted. However, if demand exceeds supply (as illustrated in Figure 22), the equilibrium price P_0 is equal to $D_{\max}$: the excess demand encourages price competition that will lead to price increases until the maximum buyer limit price is reached. This gives a rectangular PDF, illustrated in Figure 23 and stated formally as $f_3(p)$ in Equation 6.

$$
f_3(p) = \begin{cases} 0 & \text{if } p < S_{\min} \\ h_3 & S_{\min} \leq p \leq P_0 \\ 0 & p > P_0 \end{cases} \qquad (6)
$$

Let $i = P_0 - S_{\min}$ and note that $h_3 i = 1$. Substituting Equation 6 into Equation 1 gives:

$$
E(P) = \int_{S_{\min}}^{P_0} p f_3(p) \, \mathrm{d}p
$$

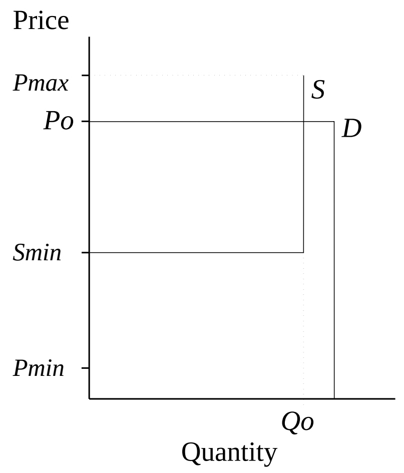

Figure 22: Flat supply and demand, with excess demand.

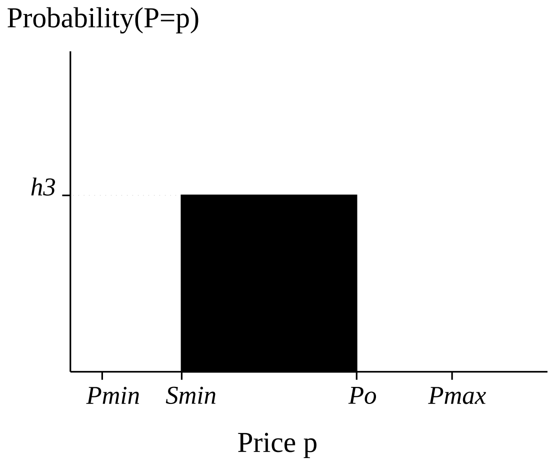

Figure 23: PDF for flat supply and demand.

$$
\begin{aligned}
&= S_{\min} + \int_0^i h_3 p \, \mathrm{d}p \\
&= S_{\min} + \left[\frac{1}{2} h_3 p^2\right]_0^i \\
&= S_{\min} + \frac{1}{2} h_3 i^2 \\
&= S_{\min} + \frac{1}{2}(P_0 - S_{\min}) \\
&= \frac{1}{2}(P_0 + S_{\min}) \quad \square \qquad (7)
\end{aligned}
$$

Hence Equation 7 demonstrates that, in situations where both supply and demand are flat, and there is excess demand, then so long as $S_{\min} \neq P_0$ the expected value $E(P)$ of transaction prices will differ from P_0.

By the same reasoning, *mutatis mutandis*, if supply and demand are flat but supply exceeds demand, then the excess supply encourages price cuts and so $P_0 = S_{\min}$; the expected value $E(P)$ differs from P_0 so long as $D_{\max} \neq P_0$, and is given by Equation 8:

$$E(P) = \frac{1}{2}(P_0 + D_{\max}) \qquad (8)$$

These four examples show that while $E(P) = P_0$ in special circumstances, in general $E(P) \neq P_0$. Similar arguments could be made for systems with discrete rather than continuous prices. The following section presents empirical evidence that supports the analytic argument developed here.

5.3 Simulation Studies

A computer simulation was written to study the behavior of ZI-C traders under different supply and demand schedules. The simulator was written in the C programming language: full details of the code are given in Appendix A, with sample input and output in Appendix B. The simulator allows for a number of ZI-C traders to be allocated roles as buyers or sellers, and to be given limit prices for their bids and offers. Results from four experiments are shown here, corresponding to the four types of supply-demand schedules examined analytically in the previous section. All experiments were run for ten trading sessions (or "days"), which continued until either eleven transactions had occurred, or no buyers or sellers were able to shout (because of the NYSE rule). In each experiment, the theoretical equilibrium values are $P_0 = 200$ and $Q_0 = 6$.

Figure 24 shows a symmetric schedule where the supply and demand curves have opposite signs but equal magnitudes: there are eleven buyers each with the right to buy one unit, and eleven sellers each with a single unit to sell, The intersection of the curves indicates $P_0 = 200$. Figure 25 shows the average transaction price of each of the ten days, over 50 experiments, with lines above and below indicating the standard deviation. As is clear, the observed average transaction price is close to 200, which is the value predicted by Equation 3.

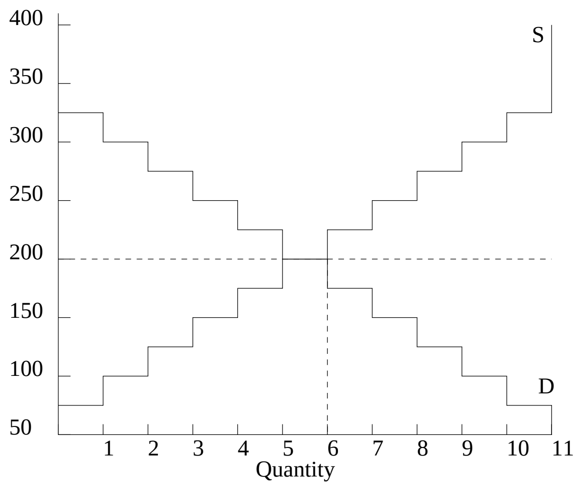

Figure 24: Symmetric supply and demand curves (gradients opposite in sign and equal in magnitude): 11 buyers and 11 sellers. Theoretical equilibrium price $P_0 = 200$; expected value of transaction prices $E(P) = 200$ from Equation 3.

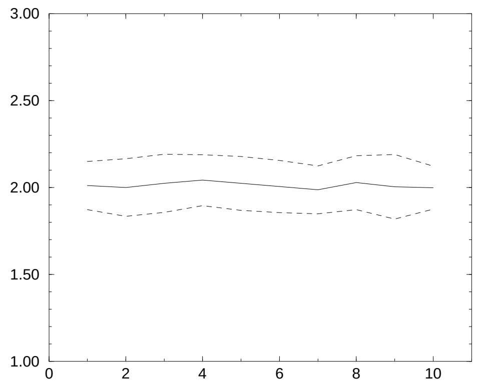

Figure 25: Mean daily transaction price, averaged over 50 ZI-C experiments, for the supply and demand shown in Figure 24: the middle line is the mean value, upper and lower lines indicates the mean plus and minus one standard deviation. Horizontal axis is day-number, vertical axis is price (divided by 100). See text for discussion.

Figure 26 shows a schedule where there is a flat supply curve. Again there are 22 ZI-C traders divided equally into buyers and sellers, each with the right to trade one unit. The intersection of the curves indicates $P_0 = 200$, but Equation 5 predicts that the observed mean value of transactions will be closer to 240. Now recall that Equation 5 is for a continuous linear demand curve, while the nonlinearities in the demand curve of Figure 26 (a consequence of having only eleven buyers) imply that the actual value of $E(P)$ may differ. By inspection, it is clear that:

$$E(P) = \left(\sum_{p=200}^{325} p \cdot g(p) \right) / \left(\sum_{p=200}^{325} g(p) \right)$$

for

$$g(p) = \begin{cases} 0 & \text{if } p < 200 \\ 5 & 200 \leq p < 225 \\ 4 & 225 \leq p < 250 \\ 3 & 250 \leq p < 275 \\ 2 & 275 \leq p < 300 \\ 1 & 300 \leq p < 325 \\ 0 & p \geq 325 \end{cases}$$

And hence the true value for the discrete nonlinear curve shown in the Figure is $E(P) = 233\frac{1}{3}$. Figure 27 then shows the average transaction price of each of the ten days, over 50 experiments, with lines above and below indicating the standard deviation. As is clear, the observed average transaction price is close to the predicted value of 233, and significantly different from the theoretical equilibrium price of 200.

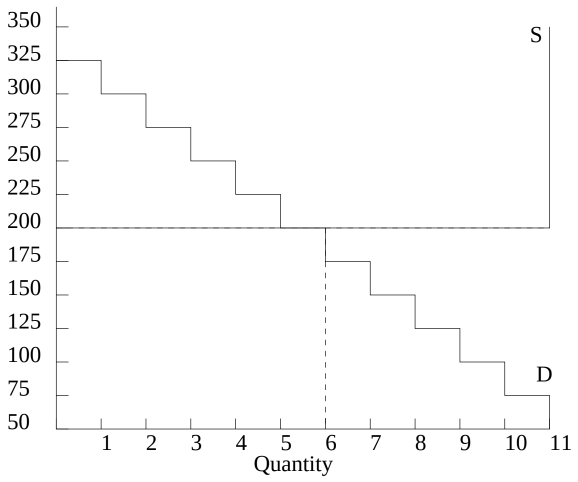

Figure 26: Flat supply: 11 buyers and 11 sellers. Theoretical equilibrium price $P_0 = 200$; expected value of transaction prices $E(P) \simeq 241$ from Equation 5, which is for a continuous linear demand curve: taking into account the discrete and nonlinear nature of the curve shown here gives $E(P) = 233\frac{1}{3}$.

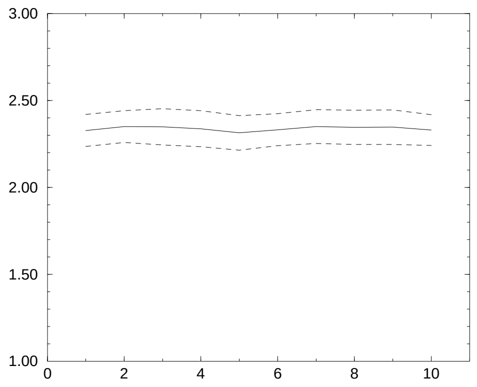

Figure 27: Mean daily transaction price, averaged over 50 ZI-C experiments, for the supply and demand shown in Figure 26: format as for Figure 25. See text for discussion.

Figure 28 shows a schedule where the supply and demand curves are both flat, and there is an excess of demand (eleven buyers and six sellers). The intersection of the curves indicates $P_0 = 200$, but Equation 7 predicts $E(P) = 125$. Figure 29 then shows the average transaction price of ZI-C traders for each of the ten days, again over 50 experiments, with lines above and below indicating the standard deviation. As is clear, the observed average transaction price is close to the predicted value of 125, and significantly different from the theoretical P_0.

Finally, Figure 30 shows flat supply and demand with excess supply (six buyers and eleven sellers). Again, $P_0 = 200$ but Equation 8 predicts $E(P) = 260$. The results from ZI-C traders are shown in Figure 31. Again, the observed mean transaction price is significantly closer to the predicted value of $E(P)$ than P_0.

These four experiments lend strong empirical support to the analytic arguments of the previous section: the empirical average transaction prices of ZI-C traders are close to the predicted

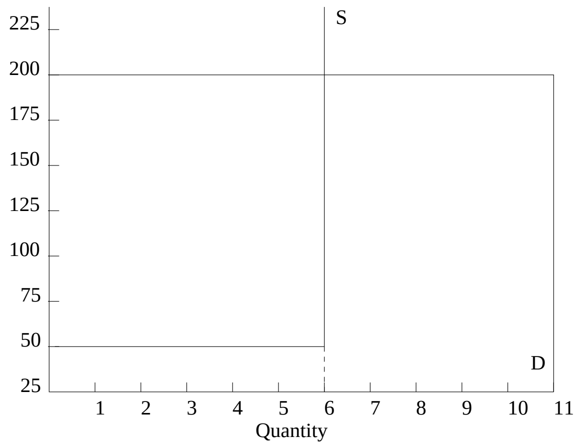

Figure 28: Flat supply and demand curves, with excess demand: 11 buyers and 6 sellers. Theoretical equilibrium price $P_0 = 200$; expected value of transaction prices $E(P) = 125$ from Equation 7.

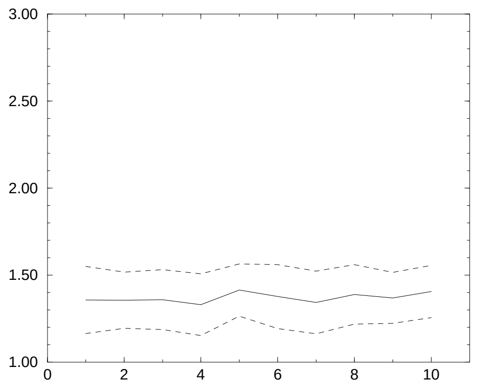

Figure 29: Mean daily transaction price, averaged over 50 ZI-C experiments, for the supply and demand shown in Figure 28: format as for Figure 25. See text for discussion.

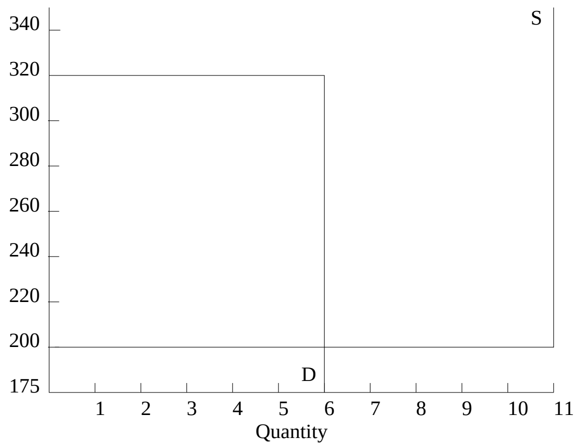

Figure 30: Flat supply and demand, with excess supply: 6 buyers and 11 sellers. Theoretical equilibrium price $P_0 = 200$; expected value of transaction prices $E(P) = 260$ from Equation 8.

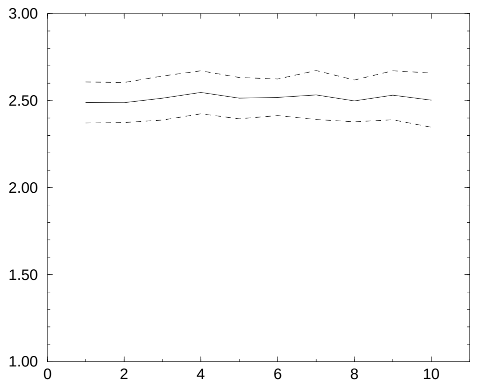

Figure 31: Mean daily transaction price, averaged over 50 ZI-C experiments, for the supply and demand shown in Figure 30: format as for Figure 25. See text for discussion.

value of $E(P)$, and in the simulations shown here the average transaction prices are only close to the theoretical equilibrium price P_0 in situations where P_0 and $E(P)$ are similar in value.

5.4 Discussion

The mathematics of Section 5.2 could be criticised for ignoring the fact that the market supply and demand curves shift after each transaction: in principle, the analysis applied only to the first transaction in each trading day. Nevertheless, there is such a good agreement between the theoretical predictions of the ZI-C traders' failure and the results from the simulations that, in practice, this criticism can be ignored.

A more subtle point is that Gode and Sunder's main claim concerned the convergence of transaction prices to equilibrium *within* a trading day: whether this happens cannot be determined from Figures 25, 27, 29, or 31, which show average transaction prices for each trading day.

To determine whether the ZI-C traders implemented here exhibit the same convergence to equilibrium as Gode and Sunder's, Figures 32 to 35 illustrate the root mean square deviation of transaction price from the equilibrium price (Smith's σ_0) calculated for each transaction sequence number. That is, for each ten-day experiment, a value $\sigma_0[1]$ is calculated from the prices of the first transaction in each of the ten days, another value $\sigma_0[2]$ is calculated from the prices of the second transaction in each day, and so on: because each day's trading with ZI-C agents is independent and identically distributed (IID), the day number is not relevant. For one experiment, this gives a vector of values $\sigma_0[i]$ where i runs from 1 to the maximum number of trades recorded in a day. Fifty experiments give fifty such vectors, and the mean and standard deviation of the values for each element of the vector are shown in Figures 32 to 35.

As can be seen, in the symmetric market of Figure 24 and the flat-supply market of Figure 26, there is a clear reduction in deviation from equilibrium as the day progresses, indicating that the transaction prices are indeed converging on equilibrium within each trading day, as observed by Gode and Sunder.

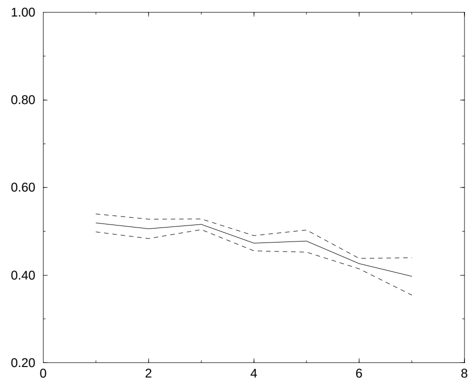

Figure 32: Horizontal axis: transaction sequence number. Vertical axis: root mean square deviation of transaction price from equilibrium price, for the symmetric market of Figure 24. Averaged over 50 experiments, each of 10 days (i.e., n=500). Solid line is mean; upper and lower dashed lines are at plus and minus one standard deviation respectively.

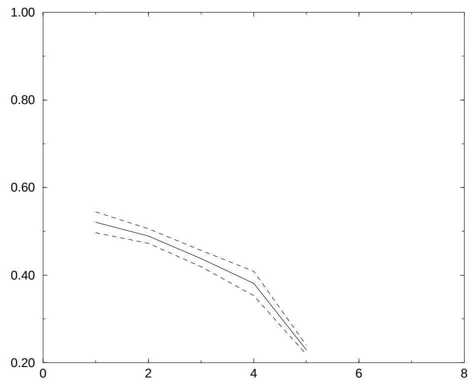

Figure 33: Horizontal axis: transaction sequence number. Vertical axis: root mean square deviation of transaction price from equilibrium price, for the flat-supply market of Figure 26. Format as for Figure 32.

However, the data in Figure 34 and 35 show that the convergence to equilibrium does not occur during trading days in the 'box design' markets of Figure 28 and 30 respectively. There is no apparent convergence to equilibrium, and this is to be expected from consideration of the market supply and demand schedules: in these two markets, all buyers have the same limit price, and all sellers have the same limit price. Therefore each *transaction* is IID, and so there can be no correlation between transaction sequence number and transaction price. Thus, in these

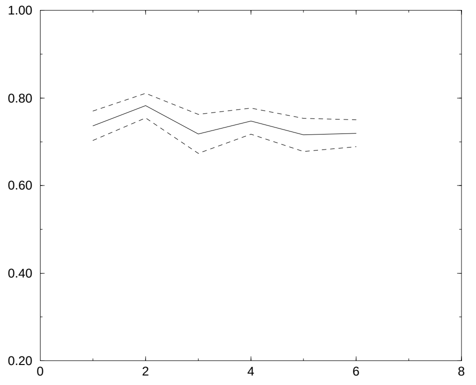

Figure 34: Horizontal axis: transaction sequence number. Vertical axis: Root mean square deviation of transaction price from equilibrium price, for the excess-demand market of Figure 28. Format as for Figure 32.

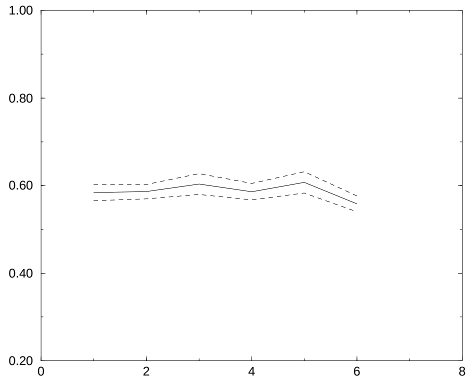

Figure 35: Horizontal axis: transaction sequence number. Vertical axis: root mean square deviation of transaction price from equilibrium price, for the excess-supply market of Figure 30. Format as for Figure 32.

markets at least, there is not even a within-day convergence toward the equilibrium price.[3]

5.5 Summary

The qualitative discussion of Section 5.1 led to the analytic demonstration that while there are conditions under which $E(P) = P_0$, in general the expected value of ZI-C transaction prices will differ from the equilibrium price. The empirical results presented in the Section 5.3 supported these theoretical predictions: in all the simulation studies, the theoretical equilibrium price $P_0 = 200$ and quantity $Q_0 = 6$, yet the mean daily trading price of ZI-C traders was only close to the theoretical P_0 when the supply and demand curves were symmetric: in the other cases, the mean ZI-C transaction prices deviated from the theoretical P_0 value by amounts predictable from the equations for $E(P)$. As the ZI-C traders are nothing more than stochastic systems generating random bids and offers, it would appear that the following hypothesis holds:

> The mean transaction price in ZI-C markets can be predicted from the expected value $E(P)$ of the probability density function (PDF) given by the intersection of the sellers' offer-price PDF and the buyers' bid-price PDF. Only in conditions where $E(P)$ is close to the theoretical equilibrium price P_0, will mean transaction prices *appear* to be close to P_0. In general, $E(P)$ and P_0 will differ, and mean transaction prices will then be at values close to $E(P)$ rather than P_0.

Furthermore, as was demonstrated in Section 5.4, although Gode and Sunder's observation of within-day convergence of transaction prices toward the equilibrium value was replicated here in

[3]It is tempting to conjecture that the speed of convergence is determined by the supply and demand schedules: Figure 33 clearly shows faster convergence than figure 32, and the lack of convergence in Figures 34 and 35 could be characterised as convergence at zero speed. Possibly this is a consequence of the differences in the supply and demand schedules of the different markets: further work would be required to determine whether there is a causal link.

two markets (Figures 24 and 26), such convergence was not observed (and indeed is theoretically impossible) in the ‘box design’ markets (Figures 28 and 30).

From this it is clear that more than zero intelligence is necessary to account for convergence to equilibrium. In the next section, trading agents with slightly more than zero intelligence are introduced, and it is demonstrated that more human-like market performance is possible with remarkably little extra brain-power.

This page is intentionally blank

6 Zero-Intelligence-Plus (ZIP) Traders

In Section 4.1, the difference between the underlying and apparent supply and demand curves was illustrated by considering the case where each trader is aiming for a particular level of profit: the profit margin determines the difference between the trader's limit price and shout-price. Intuitively at least, this has some appeal: initially, the only information known to a trader is the limit price(s) for the unit(s) the trader is entitled to sell or buy. In the absence of any other information, a profit-oriented buyer might shout a very low bid price, say $0.05, even if the buyer's limit price is $10.00: there could be a seller willing to accept the low bid, and the buyer would make a handsome profit. Similarly, a seller with a $1.00 limit price might, if no other shouts have been made, quote a price of $10.00 just to test the market. But, in a competitive market, with other traders holding roughly the same limit prices, these extreme shouts are unlikely to result in transactions. As soon as one shout has been made, every other trader in the market can use this to help determine a competitive price. As long as a trader can undercut a competitor and still make a profit, there is an incentive to do so. Thus, however extreme the initial shouts of greedy traders, there is a pressure on buyers to raise their bids, and on sellers to lower their offers: if the traders have set their profit margins too high, they will have to reduce them in order to remain competitive.

But it is also possible that a rational trader's profit margin will rise during a trading period. For instance, at the start of trading, a seller in possession of a unit with a $1.00 limit price might assume that 20% profit is a reasonable level, and so intend to shout an offer of $1.20. But, if the first few offers from competing sellers are at prices near $10.00, and there are willing buyers at these prices, the seller would be foolish to offer at $1.20: the seller's intended profit level could be increased forty-fold, and the resulting offer price of around $9.00 is still likely to be accepted by a buyer.

Thus, a plausible story can be told whereby the agents in the market adjust their profit margins up or down, on the basis of the prices of bids and offers made by the other traders, and whether those shouts are accepted, leading to deals, or ignored. Whether such a story applies to the market behavior of humans is a matter for empirical enquiry. The notes in this section report on the development of simple mechanisms where individual traders adjust their profit margins using market price information. It is demonstrated that remarkably simple adaptive mechanisms can give performance that does not suffer from the problems affecting Gode and Sunder's ZI-C traders, discussed in the previous section. Thus, these trading agents are referred to as "zero-intelligence-plus" (ZIP) traders.

Section 6.1 discusses at a qualitative level the conditions for raising or lowering a trader's profit margin. Section 6.2 then describes adaptive mechanisms that allow a trader's profit margin to alter over time. Results from simulation studies of ZIP traders operating in the markets used to illustrate the failure of ZI-C traders are then presented in Section 6.3: the code and sample input and output files for the simulator system are presented in Appendices A and B. The intention here is only to demonstrate that the simple adaptive mechanisms in ZIP traders can give results better than ZI-C traders and more similar to those of human traders. Following the comparison of ZIP and ZI-C traders, Section 6.4 presents results from using ZIP traders in experimental markets similar to those used by Smith, and illustrates the dynamics of adaptation in ZIP traders. Section 6.5 describes related research, and finally Section 6.6 discusses further issues that could be explored with ZIP traders.

6.1 Qualitative Considerations

In order to eliminate the need for sophisticated memory mechanisms, each ZIP trader alters its profit margin on the basis of four factors. The first is whether the trader is *active* in the market (i.e., still capable of making a transaction), or *inactive* (i.e., has sold or bought its full entitlement of units, and has 'dropped out' of the market for the remainder of this trading period). The three other factors all concern the last (most recent) shout: its price, denoted by q; whether it was a bid or an offer; and whether it was accepted or rejected (i.e., whether it resulted in a transaction or not). As discussed above, each trader maintains a profit margin, μ, which is multiplied by the limit price for a unit, λ, to determine the shout-price p. Increasing μ raises p for a seller and lowers p for a buyer. A ZIP buyer will, in principle, buy from any seller that makes an offer less than the buyer's current bid shout-price; similarly, a ZIP seller sells to any buyer making a bid greater than the seller's current offer shout-price.

For an inactive trader, there is little incentive to lower the profit margin: the trader has already successfully engaged in however many transactions it was entitled to. Even if this was a result of luck, it may as well wait for the luck to run out before it starts reducing its profit margin. But, if the trader drops out of the market and subsequently observes transactions occurring at prices which indicate that the inactive trader could have realised even higher profits, it should be able to raise its profit margin before the start of the next day. For these reasons, ZIP traders can raise their profit margins regardless of whether they are active or inactive, but only active traders reduce their margins.

When should a trader raise its profit margin? For a seller s_i, if the last shout resulted in a transaction, and s_i's shout-price p_i was less than the transaction price q, then the indications are that s_i could have asked an even higher price and still secured a deal, so s_i should increase its profit margin μ_i. The seller's shout price could also be increased if its shout- price equals the transaction price (i.e., if $p_i = q$), because the resultant shift in the underlying supply curve should be in the seller's favour. Thus, if $p_i \leq q$, seller s_i should increase μ_i. Similarly, a buyer b_i should raise its profit margin whenever events in the market indicate that it could buy a unit for a lower price than its current shout-price p_i: that is, whenever $p_i \geq q$.

Deciding when to lower a trader's profit margin is more difficult. If sellers compete by reducing their margins whenever a buyer makes an unsuccessful bid (below the sellers' lowest shout), the sellers would be playing into the hands of the buyers: if the buyers make a sequence of very low bids, the sellers' margins could be eroded to minimal levels while the buyers' margins remain unchanged. The reverse holds if buyers cut their profits whenever a seller makes an offer that is rejected by the buyers. Thus, if a bid is rejected, it should be the buyers that reduce their margins: in particular, any buyer who would have shouted a bid lower than the rejected value of q should reduce their margin, thereby raising their next bid. Buyers that would have shouted a bid higher than q need not reduce their margins just yet. By similar arguments, when the last shout was a rejected offer at a price q, any seller that would have shouted an offer price $p > q$ should reduce its profit margin.

But it may also be necessary for an agent to reduce its profit margin when a shout is accepted (i.e., a transaction occurs). Consider the case when a buyer makes a bid which is then accepted by a seller. If the transaction price is less than the price a seller would have shouted, then that seller should lower its profit margin because it is in danger of being undercut by the opposition (the seller that accepted the bid would have shouted a price lower than the bid). By the same reasoning, a buyer with a shout-price lower than that at which a seller's offer is accepted by some other buyer should lower its margin, so raising the price of its next bid, in order to avoid being priced out of the market.

These considerations can be summarised by the following pseudo-code:

```
• For SELLERS:

    – if (the last shout was accepted at price $q$)
    – then
        1. any seller $s_i$ for which $p_i \leq q$ should raise its profit margin
        2. if (the last shout was a bid)
           then
           1.  any active seller $s_i$ for which $p_i \geq q$ should lower its margin
    – else
        1. if (the last shout was an offer)
           then
           1.  any active seller $s_i$ for which $p_i \geq q$ should lower its margin

• For BUYERS:

    – if (the last shout was accepted at price $q$)
    – then
        1. any buyer $b_i$ for which $p_i \geq q$ should raise its profit margin
        2. if (the last shout was an offer)
           then
           1.  any active buyer $b_i$ for which $p_i \leq q$ should lower its margin
    – else
        1. if (the last shout was a bid)
           then
           1.  any active buyer $b_i$ for which $p_i \leq q$ should lower its margin
```

In order to test these qualitative bargaining mechanisms in (simulations of) real markets, it is necessary to specify how the profit margins of the buyers and sellers are raised or lowered. This requires a quantitative adaptation mechanism, discussed in the next section.

6.2 Adaptation

At a given time t, an individual ZIP trader (denoted by subscript i) calculates the shout-price $p_i(t)$ for a unit j with limit price $\lambda_{i,j}$ using the trader's real-valued profit-margin $\mu_i(t)$ according to the following equation:

$$p_i(t) = \lambda_{i,j}(1 + \mu_i(t)) \tag{9}$$

This implies that a seller's margin is raised by increasing μ_i and lowered by decreasing μ_i, with the constraint that $\mu_i(t) \in [0, \infty); \forall t$. The situation is reversed for buyers: they raise their margin by decreasing μ_i and lower it by increasing μ_i, subject to $\mu_i(t) \in [-1, 0]; \forall t$. The aim is that the value of μ_i for each trader should alter dynamically, in response to the actions of other traders in the market, increasing or decreasing to maintain a competitive match between that trader's shout-price and the shouts of the other traders. In order to do this, some form of adaptation or 'update' rule will be necessary. One of the simplest update rules in machine learning, which forms the basis of adaptation algorithms such as back-propagation in neural networks (e.g., Rumelhart, Hinton, & Williams, 1986) and reinforcement in classifier systems (e.g., Wilson, 1994, 1995), is the Widrow-Hoff "delta rule":

$$A(t + 1) = A(t) + \Delta(t) \tag{10}$$

Where $A(t)$ is the *actual output* at time t; $A(t+1)$ is the actual output on the next time-step; and $\Delta(t)$ is the change in output, determined by the product of a *learning rate* coefficient β and the difference between $A(t)$ and the *desired output* at time t, denoted by $D(t)$:

$$\Delta(t) = \beta(D(t) - A(t)) \tag{11}$$

It is clear that, if the desired output remains constant ($D(t) = k; \forall t$), the Widrow-Hoff rule gives asymptotic convergence of $A(t)$ to $D(t)$, at a speed determined by β. This adaptation method will be employed in the ZIP traders: when a trader is required to increase or decrease its profit margin (on the basis of the heuristics developed in Section 6.1), a 'target price' (denoted by $\tau_i(t)$) will be calculated for each trader, and the Widrow-Hoff rule will then be applied to take the trader's shout-price on the next time-step ($p_i(t+1)$) closer to the target price $\tau_i(t)$. Because the shout-price is calculated using the (fixed) limit price $\lambda_{i,j}$ and (variable) profit margin μ_i, it is necessary to rearrange Equation 9 to give an update rule for the profit margin μ_i on the transition from time t to $t+1$:

$$\mu_i(t+1) = (p_i(t) + \Delta_i(t))/\lambda_{i,j} - 1 \tag{12}$$

Where $\Delta_i(t)$ is the Widrow-Hoff delta value, calculated using the individual trader's learning rate β_i:

$$\Delta_i(t) = \beta_i(\tau_i(t) - p_i(t)) \tag{13}$$

All that remains is to determine how to set the target price $\tau_i(t)$. While a simple method would be to set the target price equal to the price of the last shout (i.e., $\tau_i(t) = q(t)$), this presents a significant problem. When the last shout price is very close to, or equal to, the trader's current shout price (i.e., $p_i(t) \simeq q(t)$), the value of $\Delta_i(t)$ given by Equation 13 will be very small, or zero. Thus, traders who would have shouted prices close to $q(t)$ are likely to make negligible alterations to their profit margins, and so will shout very similar prices when next given the opportunity. But in a competitive market, there is a need for the agents to be constantly testing the market, always pushing for higher margins. For example, if it happens that all traders are shouting prices in the range \$1.00 to \$1.05, the differences between their shouts and the transaction prices will never be more than a few cents, so they will hardly alter their shouts, and so the system will stabilise at this price range even if the true competitive equilibrium is at \$10.00. This sounds unlikely because, intuitively, it is desirable to have sellers always trying for higher prices and buyers always trying for lower prices. Thus, it is necessary for the target price to be different from the current shout or transaction price: for example, if a transaction occurs at \$1.00, a trader with a limit price of \$0.50 should aim for a price higher than \$1.00, while a buyer with a limit price of \$1.75 should aim for a target price lower than \$1.00.

There are many ways in which the target price $\tau_i(t)$ could be determined. In the current ZIP traders, the target price is generated using a stochastic function of the shout price $q(t)$, shown in Equation 14:

$$\tau_i(t) = \mathcal{R}_i(t)q(t) + \mathcal{A}_i(t) \tag{14}$$

Where $\mathcal{R}_i$ is a randomly generated coefficient that sets the target price *relative* to the price $q(t)$ of the last shout, and $\mathcal{A}_i(t)$ is a (small) random *absolute* price alteration (or perturbation). When the intention is to increase the dealer's shout price, $\mathcal{R}_i > 1.0$ and $\mathcal{A}_i > 0.0$; when the intention is to decrease it, $0.0 < \mathcal{R}_i < 1.0$ and $\mathcal{A}_i < 0.0$. Every time a trader's profit margin is altered, the target price is calculated using newly-generated random values of $\mathcal{R}_i$ and $\mathcal{A}_i$, which are independent and identically distributed for all traders. The use of relative increases ensures that large values of $q(t)$ are altered by greater amounts than small values of $q(t)$. For example, a shout of \$10.00 might lead to a seller's target price of \$12.50 (an absolute increase of \$2.50) while a shout of \$2.00 leads to a target of \$2.50 (an absolute increase of \$0.50), but the relative increase is the same in both cases (i.e., 25%). The use of small absolute perturbations ensures that even very small shout prices lead to targets that differ by a few cents, and can be considered as random noise in the calculation of the target price.

Finally, in many applications of the Widrow-Hoff rule where the desired output $D(t)$ varies dynamically, the learning system requires 'damping' to prevent high-frequency oscillations around $D(t)$. Consider the case where a trader's observations of the shouts and transactions in the market lead it to repeatedly increase its profit margin: if the next transaction to occur indicates that the profit margin is now too high, it may be premature to immediately reduce the margin; it might be better to reduce the *rate of increase* of the margin, rather than the margin itself. If the first indication that the margin should be reduced is reinforced by subsequent shouts or transactions, then eventually the rate of increase can take on a negative value (leading to reductions in the profit margin). Figuratively, the sequence of prices for shouts and transactions builds a "momentum" indicating which way the profit margin should be altered. This can easily be achieved by giving each trader a *momentum coefficient*, denoted by γ_i ($\gamma_i \in [0,1]$) so that if $\gamma_i = 0$ the trader takes no account of past changes when determining the next change to the value of the profit margin μ_i, but with larger non-zero values of γ_i greater emphasis is accorded to past changes. Such momentum mechanisms are often employed in back-propagation neural network learning (Rumelhart et al., 1986). Equation 15 shows the general form of the equation for momentum-based updates, with $?_i(0) = 0; \forall i$:

$$?_i(t+1) = \gamma_i ?_i(t) + (1-\gamma_i)\Delta_i(t) \tag{15}$$

Using $?_i$ in place of Δ_i in Equation 12, and defining $?_i(0) = 0; \forall i$, gives the following update rule, which is used in the ZIP traders:[4]

$$\mu_i(t+1) = (p_i(t) + ?_i(t))/\lambda_{i,j} - 1 \tag{16}$$

The behavior of groups of ZIP traders using the profit margin update rule of Equation 16 are illustrated in the following sections. In all the experiments reported there, $\mathcal{R}_i$ is uniformly distributed over the range $[1.0, 1.05]$ for price increases and over $[0.95, 1.0]$ for price decreases, giving relative rises or falls of up to 5%, and $\mathcal{A}_i$ is uniformly distributed over $[0.0, 0.05]$ for increases and $[-0.05, 0.0]$ for decreases, giving absolute alterations of up to five cents, thereby modelling a degree of uncertainty or error in the trader's formulation of the target price. The value of each trader's learning rate β_i is randomly generated when the trader is initialised, using values uniformly distributed over $[0.1, 0.5]$, and remains fixed for the duration of the experiment. Similarly, the each trader's momentum coefficient γ_i is randomly generated from a uniform distribution over $[0.2, 0.8]$ and remains constant for the duration of the experiment. Initial values for the μ_i profit margins of the traders are $[0.05, 0.35]$ for sellers and $[-0.35, -0.05]$ for buyers: that is, all traders commence each experiment with the profit margins between 5 and 35 percent.

[4] Note that when there is no momentum ($\gamma_i = 0$), Equation 16 reduces to Equation 12.

6.3 Results

To allow direct comparison, results are presented here from ZIP traders operating in the markets that were used to show the failure of ZI-C traders: the supply and demand curves for these markets were illustrated in Figure 24, 26, 28, and 30. Results showing the average of 50 runs for ZIP traders in these four markets are shown in Figures 36, 37, 38, and 39. As can be seen by comparison with the ZI-C results in Figures 25, 27, 29, and 31, the average transaction prices of the ZIP traders in these markets are much closer to the theoretical predictions than are those of the ZI-C traders. Figures 36 and 37 clearly show average transaction prices rapidly converging to the theoretical equilibrium price of \$2.00, typically within the first four trading days and remaining at that level for the remaining days, with very little variance.

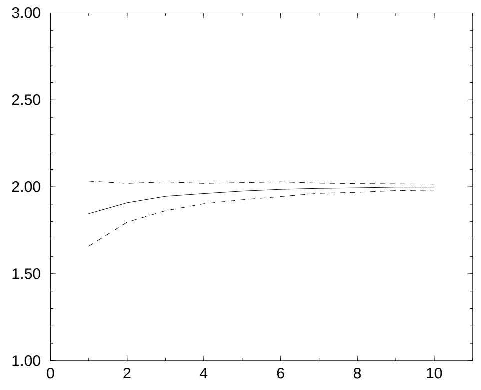

Figure 36: Mean ZIP transaction prices, averaged over 50 ZIP experiments, for the supply and demand shown in Figure 24 ($P_0 = \$2.00$): format as for Figure 25. See text for discussion.

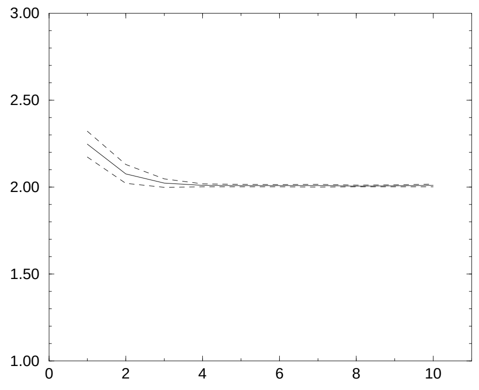

Figure 37: Mean ZIP transaction prices, averaged over 50 ZIP experiments, for the flat supply shown in Figure 26 ($P_0 = \$2.00$): format as for Figure 27. See text for discussion.

The data presented in Figures 38 and 39 are less satisfactory: the initial average transaction prices are close to those of the ZI-C traders, but this is followed by a comparatively slow (yet steady) approach to the theoretical equilibrium price, from below. To further illustrate the behavior of ZIP traders in these two markets, Figures 40 and 41 show data from experiments where there were 30 trading days, rather than 10. As is clear from these figures, the long-term tendency of the ZIP traders is towards the equilibrium price. If the various system parameters (such as the initial distributions of profit margins, and the distributions of learning rates and momentum values) were altered, faster approach to equilibrium could be demonstrated.

Similarly, the approach to equilibrium from below in Figure 36 is an artefact of the buyers and sellers having initial values of profit margin drawn from distributions over the same ranges of percentages: because the sub-marginal sellers have lower limit prices than the sub-marginal buyers, the *absolute* profit values (i.e., measured in \$) are lower for the sellers than for the buyers, and so initial transactions are more likely to occur at less-than-equilibrium prices. Again, the initial settings of the traders' parameters could be altered to eliminate this bias (i.e., give the sellers higher percentage profit margins than the buyers).

However, the intention here is not to demonstrate ZIP traders with optimal parameter-settings: rather, the data in these graphs serves to demonstrate that the simple ZIP trading strategies can readily achieve results that are impossible when using ZI-C traders, and are closer to those expected from human subjects or traditional rational-expectations theoretical predictions, with the same ZIP parameter values in a variety of market conditions. On these grounds at least, the minimally adaptive ZIP traders represent a significant advance on the work of Gode and Sunder.

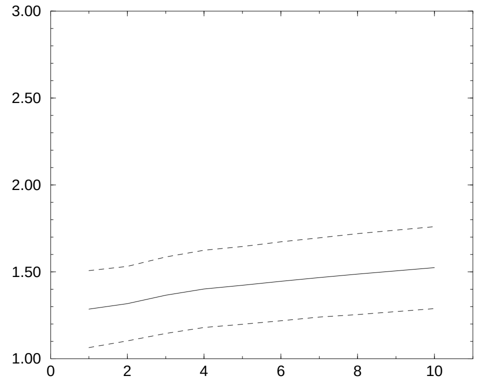

Figure 38: Mean ZIP transaction prices, averaged over 50 ZIP experiments, for the flat supply and demand, with excess demand, shown in Figure 28 (P_0 = \$2.00): format as for Figure 29. Trading for 10 days. See text for discussion.

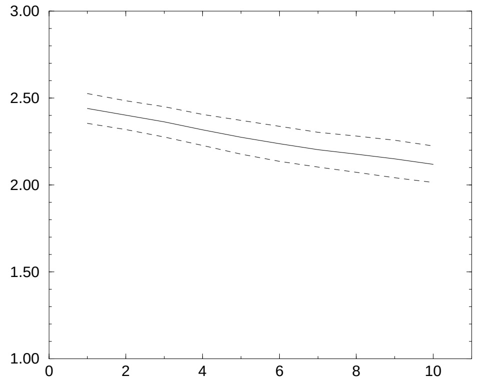

Figure 39: Mean ZIP transaction prices, averaged over 50 ZIP experiments, for the flat supply and demand, with excess supply, shown in Figure 30 (P_0 = \$2.00): format as for Figure 31. Trading for 10 days. See text for discussion.

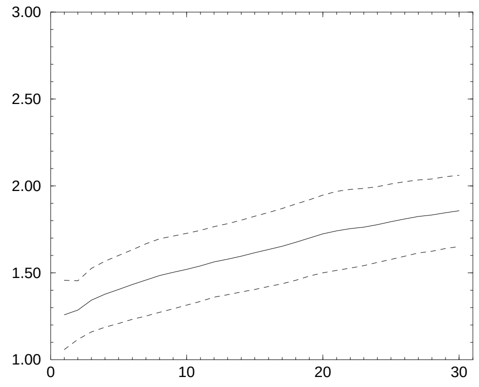

Figure 40: Mean ZIP transaction prices, averaged over 50 ZIP experiments, for the flat supply and demand, with excess demand, shown in Figure 28 (P_0 = \$2.00): format as for Figure 29. Trading for 30 days. See text for discussion.

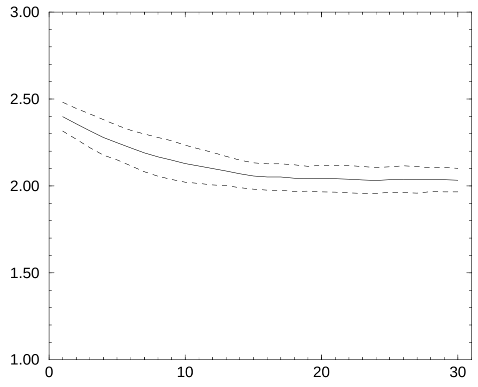

Figure 41: Mean ZIP transaction prices, averaged over 50 ZIP experiments, for the flat supply and demand, with excess supply, shown in Figure 30 (P_0 = \$2.00): format as for Figure 31. Trading for 30 days. See text for discussion.

It is also possible to plot Smith's measure of allocative efficiency and Gode and Sunder's measure of profit dispersion for the ZIP traders. As with the ZI-C traders, measures of allocative efficiency for ZIP traders are typically very high (often averaging 100%), and so are not plotted here. However, plots of profit dispersion are more revealing: average profit dispersion values for both ZI-C and ZIP traders in the four markets introduced in Figures 24, 26, 28, and 30 are shown in Figures 42 to 45 respectively.

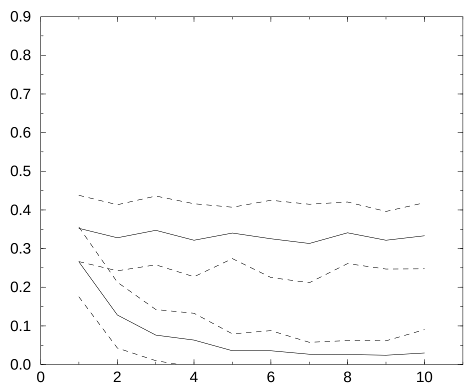

Figure 42: Mean ZI-C and ZIP profit dispersion levels for the market of Figure 24, over 10 days. The ZI-C traders show a near-constant mean profit dispersion around 0.35, while the mean dispersion for ZIP traders falls rapidly and stabilises at values less than 0.05.

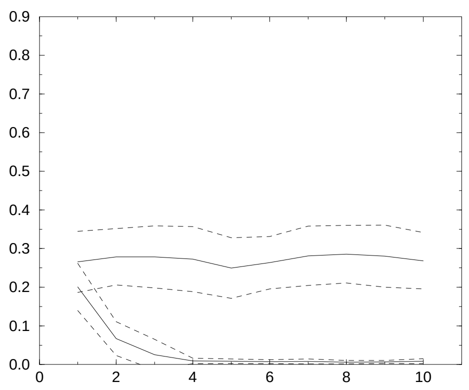

Figure 43: Mean ZI-C and ZIP profit dispersion levels for the market of Figure 26, over 10 days. The ZI-C traders show a near-constant mean profit dispersion around 0.25, while the mean dispersion for ZIP traders falls rapidly and stabilises at values close to 0.01.

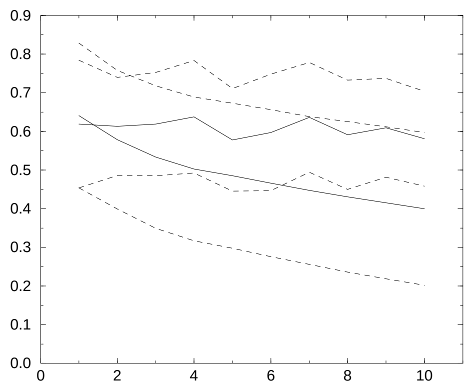

Figure 44: Mean ZI-C and ZIP profit dispersion levels for the market of Figure 28, over 10 days. The ZI-C traders show a near-constant mean profit dispersion around 0.6, while the mean dispersion for ZIP traders falls gradually from approximately 0.65 to near 0.4.

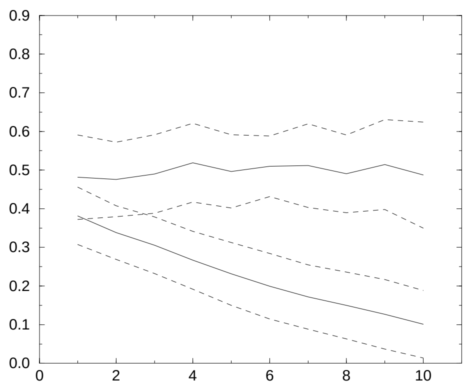

Figure 45: Mean ZI-C and ZIP profit dispersion levels for the market of Figure 30, over 10 days. The ZI-C traders show a near-constant mean profit dispersion around 0.5, while the mean dispersion for ZIP traders falls steadily from less than 0.4 to around 0.1

As can be seen from the profit dispersion figures, in all cases the final (Day 10) profit dispersion is significantly less for ZIP traders than for ZI-C traders. In Figures 42 and 43 the ZIP profit dispersion falls sharply over the first four days and then levels out to a roughly constant value; in Figures 44 and 45 the fall is less dramatic but could, presumably, be made more rapid by appropriate alteration of the parameter-settings, as was discussed previously. But, to reiterate, the intention here is to show the performance of ZIP traders with identical parameter settings in a variety of markets, rather than with parameter settings tuned for each market. As was discussed in Section 4.2, Gode and Sunder (1993) note that the ZI-C profit dispersion levels are lower than those of the ZI-U traders but appreciably higher than those of human traders. As is demonstrated in Figures 42 to 45, ZIP traders rapidly adapt to give profit dispersion levels that are in some cases approximately a factor of ten less than those of ZI-C traders. On this basis, it seems safe to claim that the performance of the ZIP traders in the experimental markets used here is significantly closer to that of human traders than is the performance of ZI-C traders.

6.4 Discussion

In addition to comparing the behavior of ZIP and ZI-C traders, we can also compare the behavior of ZIP traders to Smith's results from human subjects. The symmetric market in Figure 24 is clearly comparable to the markets used in some of Smith's early experiments (e.g., Charts 1 to 3 in Smith (1962): Chart 1 was shown in Figure 6). The flat-supply market in Figure 26 is comparable to Smith's (1962) Chart 4 (shown in Figure 7), and the excess-demand market in Figure 28 is comparable to Smith's (1962) Chart 6 (Figure 8). In particular, Smith notes that in his excess-demand market, "... The approach to equilibrium is from below, and the convergence is relatively slow.": both of these qualities are exhibited by the ZIP trader results in Figures 38 and 40 but not the ZI-C trader results in Figure 29.

In the ZIP experiments shown so far, the supply and demand schedules have remained fixed for the duration of the experiment. However, as was noted in Section 4.1, Smith (1962) also experimented with dynamic changes in supply or demand: in some of his experiments, at the end of a trading day a new set of limit prices was distributed to the buyers, sellers, or both. Typically, the human traders would adapt, converging to the new market equilibrium values. This rapid, robust, and decentralized adaptation is one of the attractions of using the continuous double auction as a market organisation. Thus, it is important to explore the behavior of ZIP traders when supply or demand alter (either increase or decrease): for ZIP traders to be of genuine use in applications of market-based control or internet-based commerce, they should exhibit smooth and fast convergence to the new equilibrium that results from shifts in supply or demand.

Figure 46 shows a transaction-price time-series from one experiment which uses the symmetric market of Figure 24 for the first ten days. At the end of Day 10, an increase in demand is imposed: the demand curve is shifted upwards by adding \$0.50 to each buyer's limit price (P_0 increases to \$2.25), and the experiment continues for another five days. Figure 47 shows the average results from 50 such experiments. Similarly, Figure 48 shows transaction prices from one experiment where the symmetric market of Figure 24 is again used for the first ten days, but an increase in supply is then imposed by subtracting \$0.50 from each seller's limit price (P_0 decreases to \$1.75) and trading continues for another five days. These figures clearly demonstrate that groups of ZIP traders are capable of rapidly adjusting to new equilibrium values resulting from changes in supply or demand.

In Smith's (1962) paper, one experiment examined the effects of a different market structure, where only sellers were allowed to shout offers: buyers were not allowed to shout bids, but could passively observe the prices offered by the sellers (Smith's results from this experiment were shown in Figure 9). Each buyer therefore had the privilege of being able to ignore offer prices that were "too high" and accept prices that were within their range, without giving any indication of

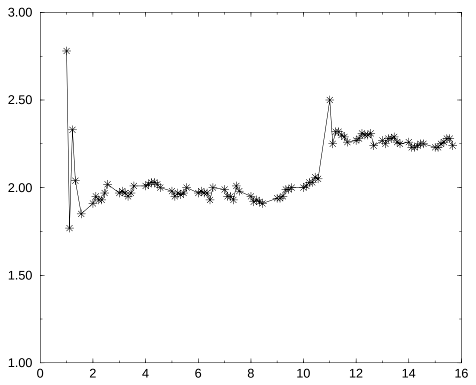

Figure 46: Transaction-price time series for one experiment with a sudden increase in demand. Initial market is illustrated in Figure 24 (P_0 = \$2.00). After 10 trading days, demand is increased (P_0 = \$2.25) and the experiment continues for another 5 days. See text for discussion.

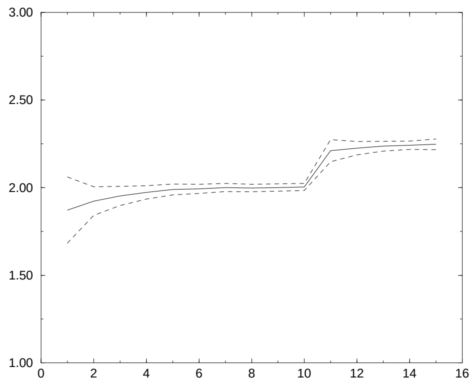

Figure 47: Mean ZIP transaction prices, averaged over 50 increased-demand experiments.

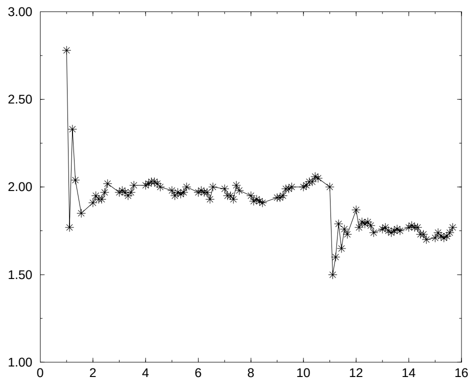

Figure 48: Transaction-price time series for one experiment with a sudden increase in supply. Initial market is illustrated in Figure 24 (P_0 = \$2.00). After 10 trading days, supply is increased (P_0 = \$1.75) and the experiment continues for another 5 days. See text for discussion.

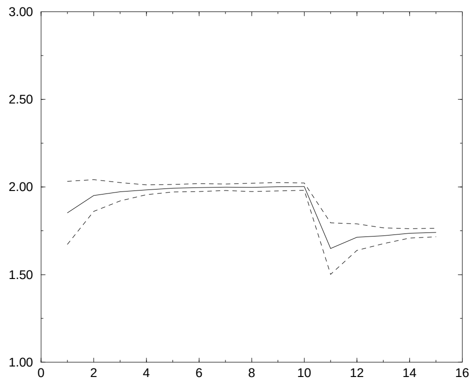

Figure 49: Mean ZIP transaction prices, averaged over 50 increased-supply experiments.

their limit prices. Smith proposed this as an approximation to an ordinary retail market, where sellers bear the responsibility of advertising their prices and buyers decide whether to buy or not without entering into any kind of bargaining or haggling process. Smith's comments on his expectations and actual results for this experiment are illuminating:

> "Since sellers desire to sell at the highest prices they can get, one would expect the offer prices to be high, and, consequently, one might expect the exchange [i.e., transaction] prices to show a persistent tendency to remain above the predicted equilibrium. The result was in accordance with this crude expectation in the first market period [i.e., day] only.... Since sellers only were making offers, the prices tended to be very much above equilibrium. Five of these offers were accepted at prices ranging from \$2.69 to \$2.80... The competition of sellers pushed the offer prices lower and the remaining buyers made contracts at prices [of \$2.35, \$2.00, and \$2.00]. The early buyers in that first market period never quite recovered from having subsequently seen exchange prices fall much below the prices at which they had bought. Having been badly fleeced, through ignorance, in that first trading period, they refrained from accepting any high price offers in the remaining three periods of the test. This action, together with seller offer price competition, kept exchange prices at levels persistently below equilibrium for the remainder of [the experiment]." Smith (1962).

While it is not immediately clear how Gode and Sunder's ZI-C trader experiments could be modified to allow for such studies, the ZIP traders can be used in a straightforward copy of Smith's experimental retail market. The supply and demand curves for the market are shown in Figure 50, and the mean daily transaction prices of ZIP traders (in 50 experiments, with the same parameter values as used in the previous ZIP experiments) are shown in Figure 51. As can be seen, the average transaction prices are typically less than \$2.00 (significantly below the theoretical equilibrium price of \$2.25). There also appears to be little or no convergence towards equilibrium, or reduction in variance. The apparent lack of convergence or reduction in variance can be better understood by examining individual price trajectories: Figures 52 to 55 show time-series of the transaction prices in four individual experiments using ZIP traders in the market of Figure 50. As can be seen, in all four experiments the market converges to a fairly constant transaction price by Day 4, but the value converged on can vary: in Figures 52 to 54, all trades on Day 10 are within \$0.15 of the theoretical equilibrium, while in Figure 55 no trade is less than \$0.40 off the equilibrium price. As is clear in Figure 51, the price converged on is, on average, significantly less than the theoretical equilibrium: qualitatively, this result agrees with Smith's (1962) observations of human subjects in his experimental 'retail markets'.

Of these four single experiments, the price series in Figure 52 most closely resembles that of Smith's subjects: only three trades occur at transaction price more than a few cents above the equilibrium price; while many more occur at prices lower than equilibrium, which is approached *very* slowly, from below. Whether this is due to early trades at high prices preceding a series of low-price trades that induce a resistance to higher prices in 'fleeced' traders requires a more detailed examination of the dynamics of individual experiments. In Figure 56, text output from Day 1 is shown: in the first four trades, sellers announce a price and one or more buyers are willing to buy at that price (the buyer who gets the deal is chosen at random from those that are willing). In the fifth trade, Seller 10 makes an offer of \$3.52 which is ignored by the buyers: Seller 9 then offers at \$3.51; this is also ignored and Seller 9 offers again at \$3.50, which is again ignored; Seller 5 then offers at \$2.37, which is taken up by Buyer 0. For sixth trade, there is a sequence of 33 ignored offers, which ends when Seller 4 makes an offer of \$2.12 (having

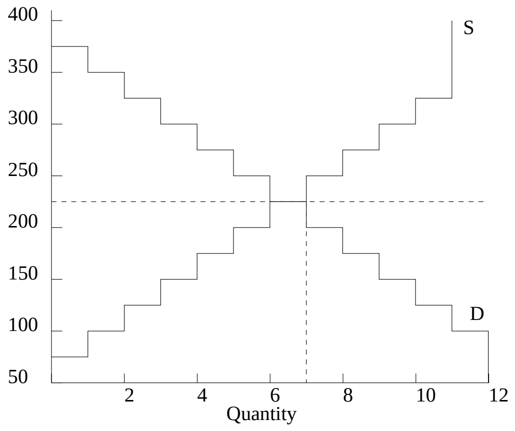

Figure 50: Supply and demand for market where only sellers shout: 12 buyers and 11 sellers. Theoretical equilibrium price $P_0 = 2.25$; quantity Q_0=7.

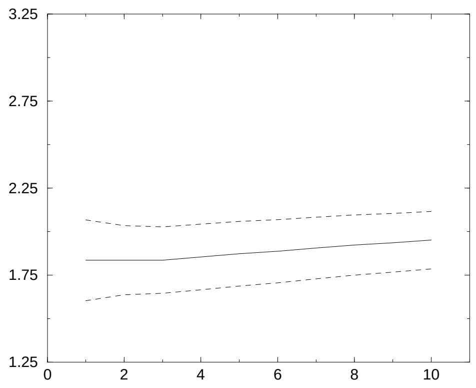

Figure 51: Mean ZIP transaction prices, averaged over 50 experiments, for the market of Figure 50. Format as for Figure 25

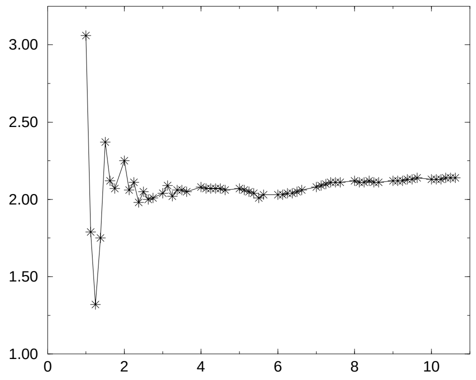

Figure 52: Transaction-price time series for one experiment of the market of Figure 50

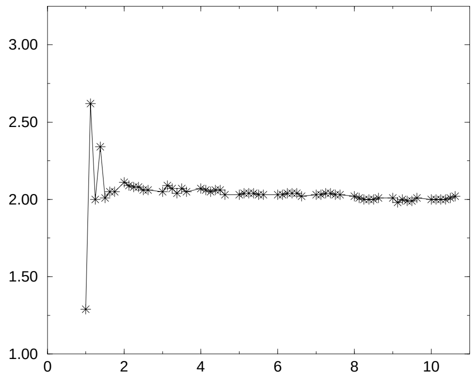

Figure 53: Transaction-price time series for one experiment of the market of Figure 50.

previously offered \$2.40, \$2.22, and \$2.16). For the seventh, there are 49 ignored offers before Seller 3 finally drops the offer price to \$2.07, and a deal is done. In the eighth trade, 100 shouts fail to find a taker, and the first trading day ends.

The long sequences of ignored shouts can be made less likely by employing the NYSE rule: running the same experiment but with the improvement rule enforced leads to a termination of the first day after the fifth transaction: in the sixth trade, Seller 3 offers at \$2.21 but this offer is ignored and no other active sellers are able to make a better offer and so (because of the NYSE rule) the first trading day is ended. Using the NYSE rule in this market gives a transaction-price time series broadly similar to that of Figure 52, although in other markets it can impair the equilibration of ZIP traders, because it reduces the number of shouts they are exposed to and therefore slows the adaptation process.

The effects this sequence of accepted and ignored offers has on the profit margins of the ZIP

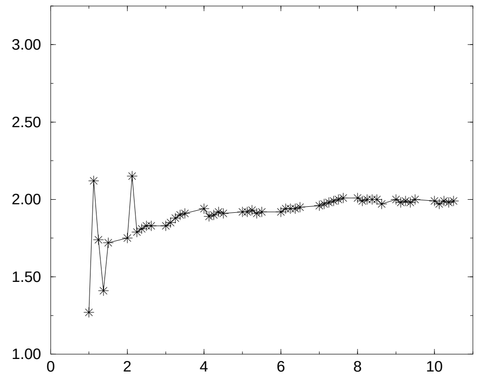

Figure 54: Transaction-price time series for one experiment of the market of Figure 50.

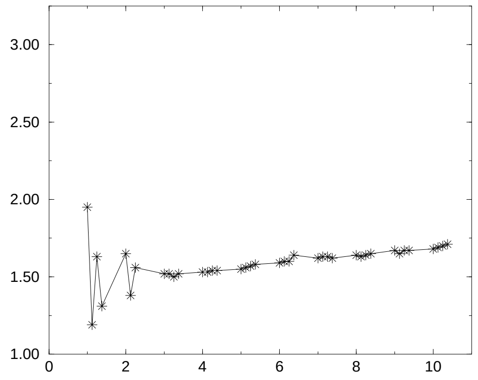

Figure 55: Transaction-price time series for one experiment of the market of Figure 50.

buyers and sellers is illustrated in Figure 57, which shows the bid-and-offer arrays at the start of Day 1 and at the start of Day 2. As can be seen, the apparent supply and demand curves have altered significantly. For intra-marginal units, the traders have increased their profit margins, flattening the supply and demand curves and bringing them closer together, thereby reducing the apparent surplus. For extra-marginal units, the traders have decreased their profit margins, again lessening the distance between the curves.

To better illustrate the alterations in the bid-and-offer arrays between the two states shown in Figure 57, Figure 58 shows the temporal progression of the arrays after each attempt to trade in Day 1. As can be seen from the graphs labelled E to H, after four transactions the apparent supply and demand curves do not intersect, and so there is no equilibrium price or quantity. This gives rise to sequences of ignored shouts (5 before Figure 58E, 33 before Figure 58F, 49 before Figure 58G, and 100 before Figure 58H), which in turn lead to alteration of the traders' profit margins, thereby altering the apparent supply and demand so that eventually an intersection does occur, and then a transaction can take place. Typically, as soon as the apparent supply and demand curves intersect, two traders make a deal and leave the market, and in doing so they alter the apparent supply and demand back to a state where no equilibrium is indicated.

Figure 59 shows the bid-and-offer arrays at the start of each subsequent day in the experiment. As is clear, although the rank ordering of the traders varies as they alter their prices up or down by a few cents, there is very little change in the overall shape of the bid-and-offer arrays after Day 3. The fact that in this experiment the market converges on transactions around \$2.12 (i.e., less than the theoretical equilibrium price of \$2.25) is not a problem: it is consistent with Smith's (1962) results from his experiment with human subjects, where transaction prices also converged to a stable below-equilibrium level; and may be a consequence of using such a one-sided market structure (i.e, this is *not* a continuous double auction).

Significantly, explanations of why the data in Figure 51 converges on a stable price below equilibrium cannot rely on folk-psychological notions such as 'badly fleeced' buyers resisting price increases: by specifying and observing simple synthetic trading agents, it is possible to demonstrate the same overall market behavior without relying on abstract or vague descriptions of the mental states of the participants in the market. In this sense, the work described here is similar to other work in cognitive science that is justified by the principle that it can be more

```
day 0 trade 1
Seller  7 offers at 3.060 (reward=0.560) 1 traders willing to deal
Seller  7 sells to Buyer 1 (reward=0.440)

day 0 trade 2
Seller  2 offers at 1.790 (reward=0.540) 5 traders willing to deal
Seller  2 sells to Buyer 3 (reward=1.210)

day 0 trade 3
Seller  0 offers at 1.320 (reward=0.570) 8 traders willing to deal
Seller  0 sells to Buyer 8 (reward=0.430)

day 0 trade 4
Seller  1 offers at 1.750 (reward=0.750) 6 traders willing to deal
Seller  1 sells to Buyer 6 (reward=0.500)

day 0 trade 5
Seller 10 offers at 3.520 (reward=0.270) No willing takers (fails=1)
Seller  9 offers at 3.510 (reward=0.510) No willing takers (fails=2)
Seller  9 offers at 3.500 (reward=0.500) No willing takers (fails=3)
Seller  5 offers at 2.370 (reward=0.370) 1 traders willing to deal
Seller  5 sells to Buyer 0 (reward=1.380)

day 0 trade 6
Seller  3 offers at 2.210 (reward=0.710) No willing takers (fails=1)
Seller  6 offers at 2.520 (reward=0.270) No willing takers (fails=2)
Seller  6 offers at 2.530 (reward=0.280) No willing takers (fails=3)
Seller  8 offers at 2.930 (reward=0.180) No willing takers (fails=4)
Seller 10 offers at 3.350 (reward=0.100) No willing takers (fails=5)
Seller  9 offers at 3.080 (reward=0.080) No willing takers (fails=6)
Seller  8 offers at 2.820 (reward=0.070) No willing takers (fails=7)
Seller  9 offers at 3.040 (reward=0.040) No willing takers (fails=8)
Seller  9 offers at 3.010 (reward=0.010) No willing takers (fails=9)
Seller  4 offers at 2.400 (reward=0.650) No willing takers (fails=10)
...
Seller  4 offers at 2.220 (reward=0.470) No willing takers (fails=15)
...
Seller  4 offers at 2.160 (reward=0.410) No willing takers (fails=24)
...
Seller  8 offers at 2.780 (reward=0.030) No willing takers (fails=32)
Seller  9 offers at 3.010 (reward=0.010) No willing takers (fails=33)
Seller  4 offers at 2.120 (reward=0.370) 1 traders willing to deal
Seller  4 sells to Buyer 4 (reward=0.630)

day 0 trade 7
Seller 10 offers at 3.350 (reward=0.100) No willing takers (fails=1)
...
Seller  8 offers at 2.780 (reward=0.030) No willing takers (fails=49)
Seller  3 offers at 2.070 (reward=0.570) 1 traders willing to deal
Seller  3 sells to Buyer 2 (reward=1.180)

day 0 trade 8
...
Seller  8 offers at 2.760 (reward=0.010) No willing takers (fails=100)
```

Figure 56: Text output showing shouts and deals for Day 1 in the experiment of Figure 52. Much text has been deleted to increase clarity.

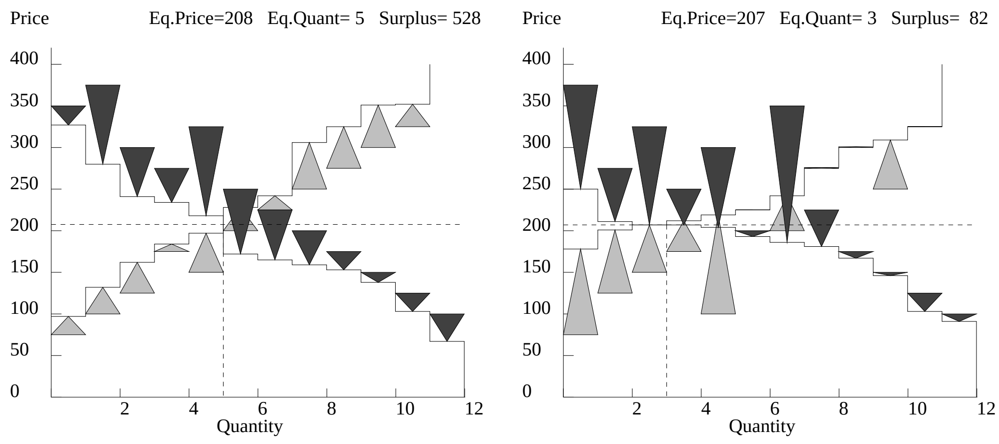

Figure 57: Bid-and-offer arrays in the experiment of Figure 52. Limit and shout prices are indicated using the triangles introduced in Figure 5. Left: at the start of Day 1. Right: at the start of Day 2.

fruitful and more parsimonious to attempt an understanding of how some behavior is generated by *synthesising* an artificial system that exhibits that behavior, rather than by *analysing* a natural system that exhibits the same behavior (see e.g., Braitenberg (1984) and Cliff and Noble (1997)).

Thus, although the discussion here has demonstrated that ZIP traders can give results qualitatively similar to those of humans in 'retail market' experiments (and that ZI-C traders probably could not), that is a relatively minor point. Perhaps of more general significance is that with synthetic adaptive agents it is possible to record all manner of significant variables, both internal and external to the agent, and to visualise them in styles such as those shown in Figures 51 to 59. And this is from just one experiment, which took less than five seconds to run on a medium-power workstation (a Sun Sparc20). Clearly, tens or hundreds of thousands of experiments can be run with artificial agents in the time it takes one experiment to be conducted with human subjects. This is not necessarily an advantage: each experiment has the potential to generate masses of data; managing, visualising, and analysing the data to arrive at meaningful conclusions could present serious problems, and should be noted as a topic for further work. Other further directions in which this work could be taken are discussed Section 6.6: before that, Section 6.5 describes related work.

6.5 Related Work

As was noted earlier, Gode and Sunder's work on ZI traders has been cited approvingly in a number of texts discussing continuous double auctions. Despite this, there appear to be very few papers that are comparable to the work described here: I know of no other critiques of Gode and Sunder's work, and have found only two papers that describe artificial trading agents similar to the ZIP traders developed here. These two papers are by Easley and Ledyard (1992) and Rust et al. (1992).

Easley and Ledyard (1992) consider several theories for price formation and equilibration, attempting to explain how human traders converge to equilibrium. They introduce a mathematical notation which they use to describe specific hypotheses concerning trading strategies and equilbration in double-auctions; a number of analytic proofs then lead to three specific predictions, which they test by comparison to data from human experiments. Their trading strategies

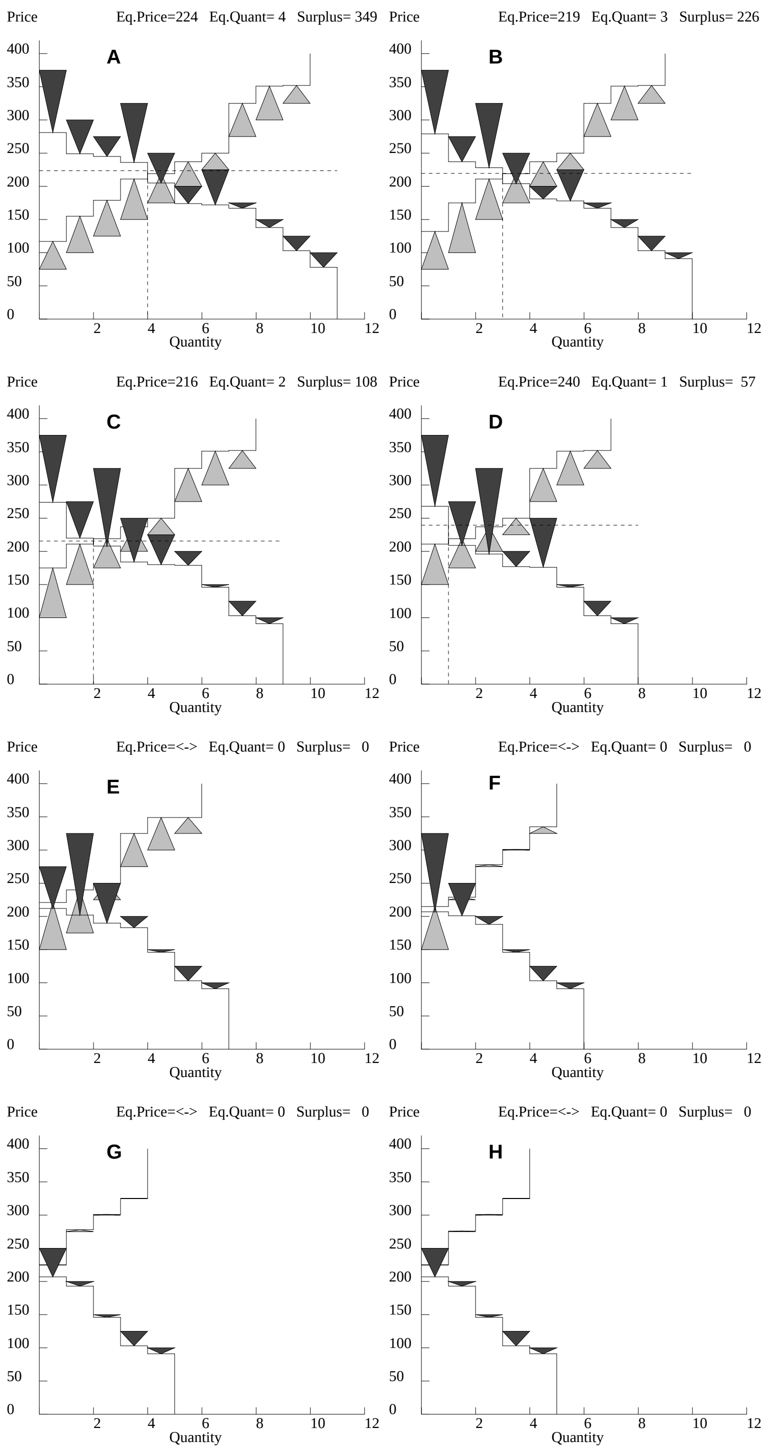

Figure 58: Temporal progression of bid-and-offer arrays for days 1 to 2 in the price series shown in Figure 52. Each graph shows the bid-and-offer arrays of the active traders after a transaction: A is after the first transaction; B is after the second transaction; And so on until H which is after the eighth (end of Day 1).

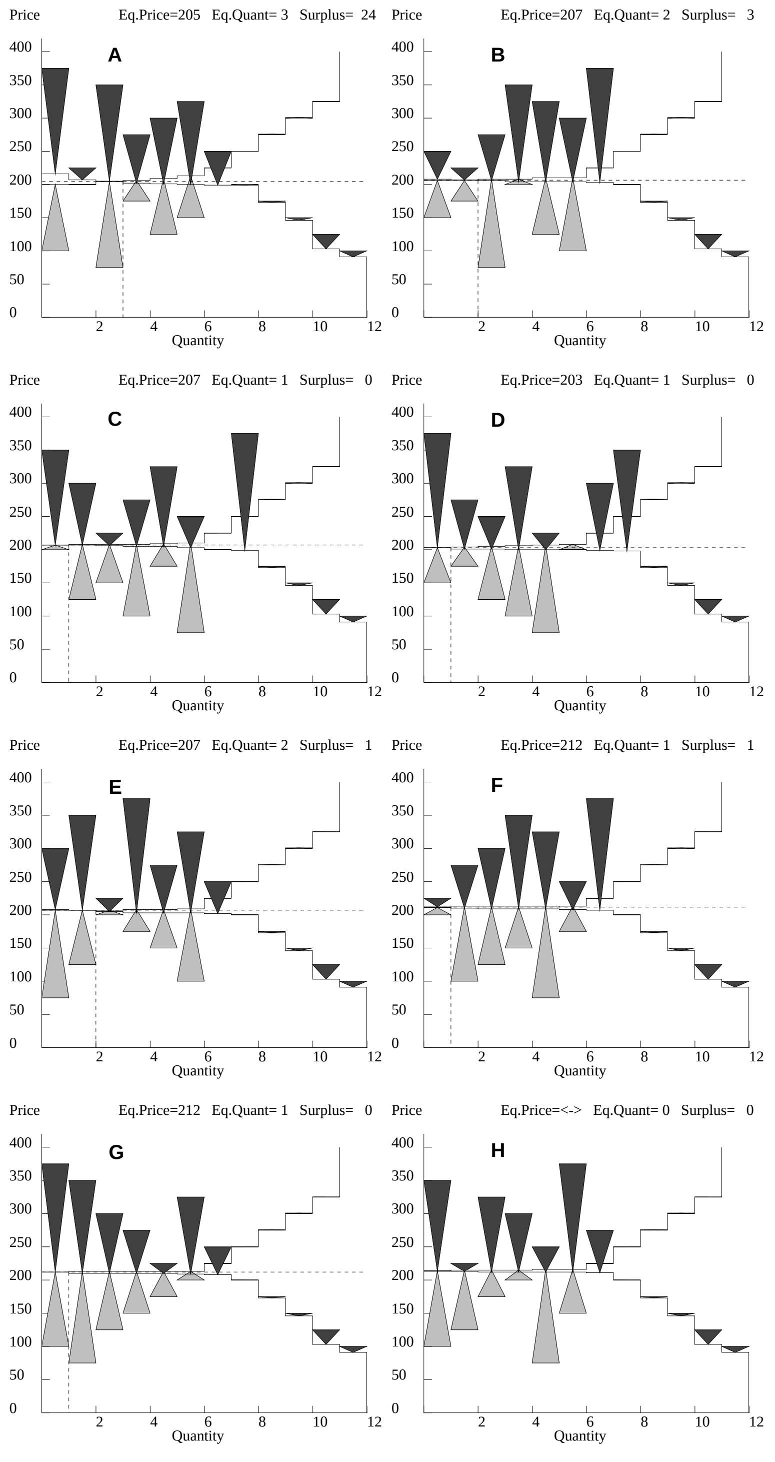

Figure 59: Temporal progression of bid-and-offer arrays for days 3 to 10 in the price series shown in Figure 52. Each graph shows the bid-and-offer array at the start of a day's trading: A is day 3; B is day 4; and so on until H which shows the start of day 10. See text for discussion.

are simple mechanisms which rely on a memory of data from past trading days. Specifically, Easley and Ledyard's trading strategies use the following information: the lowest-priced offer or contract in the previous day's trading; the highest-priced bid or contract in the previous day's trading; the most recent bid, offer and contract prices in the current day's trading; the time remaining to the end of the current trading day; and an indicator of whether the agent has traded in the current day (their traders are designed to trade only one unit per day, so this indicator is similar to the way in which the ZIP traders cease to be active once they have traded all their entitlement; however, ZIP traders may enter into more than one transaction per day). Clearly, Easley and Ledyard's trading strategies could require more memory than a ZIP trader, and also their strategy is only fully effective after the first day of trading; yet it is often on the first day that the most significant shifts in behavior occur. Their analysis relies on a simplifying assumption that is questionable in practice: they assume that, when more than one trader is interested in a transaction, the buyer with the highest shout price or the seller with the lowest shout price is guaranteed the deal (Easley & Ledyard, 1992, p.70). Despite (or possibly because of) this, several of the experimental observations they present contradict their theoretical predictions. Furthermore, as Easley and Ledyard (1992, p.87) note, their theory does not apply to experiments in which one side of the market is not allowed to bid or offer (e.g., 'retail' markets), and it doesn't predict the effects of shifts in supply and demand curves. As was demonstrated in Section 6.4, ZIP traders can give human-like equilibration in such situations: for this reason, it would seem that ZIP traders are an advance on the work of Easley and Ledyard (1992).

Rust et al. (1992) report on a series of experimental economics tournaments they organised, where other researchers were invited to submit software agents that would compete against one another in a simplified double auction. The double auction was simplified by synchronizing it into a two-stage process that was iterated several times per trading day. In the first stage, all traders simultaneously shout a bid or offer, and these shouts are distributed to all traders. In the second stage, the trader with the highest current bid and the trader with the current lowest offer are given the option of entering into a transaction: they can either agree a deal, or refuse. In addition to the array of bids and offers, each trader has access to public information which includes the number of buyers; the number of sellers; the identities of the traders; the number of rounds (experiments), periods (days per experiment) and timesteps (iterations per day); the number of units each agent will have; and the distribution from which the unit limit prices are generated (Rust et al., 1992, p.164). The trading agents were allowed to be both buyers and sellers, although some researchers submitted seller-only or buyer-only strategies. A number of tournaments were held, and the different strategies were ranked in order of the profits they generated. Few details are given of the specifications of the different strategies, so a detailed comparison with ZIP traders is difficult. However, a key difference between these tournaments and the ZIP (and ZI-C) experiments is that the tournaments involved *heterogeneous* groups of traders. Traders with radically different strategies could compete in the same market, and much of the focus in Rust et al. (1992) is on the way in which the different trading strategies interacted, both with a fixed number of different strategies and in 'evolutionary' tournaments where the relative numbers of the different trading strategies altered over time, so more profitable strategies became more numerous than less profitable ones. The 'population dynamics' of the tournaments occupy much of the discussion:

> "We find that the top-ranked programs yield a fairly "realistic" working model of a [double auction] market in the sense that their collective behavior is consistent with the key "stylized facts" of human experiments. We also find that a very simple strategy is a highly effective and robust performer in these markets. This strategy was

able to outperform more complex algorithms that use statistically based predictions of future transaction prices, explicit optimizing principles, or sophisticated "learning algorithms". The basic idea behind the approach can be described quite simply: *wait in the background and let others do the negotiating, but when bid and [offer] get sufficiently close, jump in and "steal the deal"*. However, the results of our evolutionary tournaments show that when too many other traders try to imitate this strategy, market efficiency can fall precipitously.... Specifically, if too many traders "wait in the background", little information is generated until just before the end of the trading period. This tends to produce "closing panics" as traders rush to unload their [units] in the final seconds of the trading period, resulting in failure to execute all potentially profitable transactions." (Rust et al., 1992, p.157, original emphasis).

Thus, there is no focus in Rust et al. (1992) on explicit critiques of Gode and Sunder's ZI traders, or on exploring the behavior of homogeneous groups of traders in particular market environments such as the symmetric, flat supply, excess-supply 'box', excess-demand 'box', increased-demand symmetric, increased-supply symmetric, and 'retail' markets used with ZIP traders in Sections 6.3 and 6.4. Furthermore, the reproductive success of the "wait in the background" trader strategy indicates that the 'evolutionary' tournaments can favour trading strategies that, when used to form homogeneous groups of traders, can give rise to market dynamics that are manifestly ill-suited to applications in market-based control or internet-based commerce.

The new scientific field of *artificial life* is often characterised as the study of complex adaptive systems. In brief, such systems typically exhibit complex coherent global behavior arising from the interaction of groups of components which are individually simple in comparison to their global behavior. A number of major scientific problems fall within this category. Examples include: the origins of life (in the form of self-sustaining or 'autocatalytic' cyclic chemical chain reactions) from a pre-biotic chemical 'soup'; the co-operative and competitive co-evolution of self-replicating and self-regulating organisms; the emergence of coherent patterns of activity from the asynchronous firing of groups of nerve cells; co-ordinated group motion such as flocking in birds or schooling in fish; and so on. Studies in artificial life typically involve computational models or simulations, studying models that are too complex to yield to analytic approaches. Given this focus, the convergence to equilibrium of groups of traders operating in market-based environments would seem to be a natural candidate for artificial life research.

However, the international artificial life journal and conference proceedings show a distinct lack of such research. While there is a small core of work on the *iterated prisoner's dilemma*, a classic game-theory problem in which the emergence of cooperative behavior among non-altruistic agents can be explored, and of direct relevance to oligopolistic markets (see, e.g., Axelrod (1984), Stanley, Ashlock, and Tesfatsion (1993) Batali and Kitcher (1994), and May, Bohoeffer, and Nowak (1995)), I know of only two papers published in the artificial life literature that explicitly study market trading strategies: Nottola, Leroy, and Davalo (1991) and de la Maza and Yuret (1994). Both of these papers report on the application of simple evolutionary adaptation methods to optimize simple trading strategies for speculative markets, and both set the equilibrium price via a centralised process that collects prices from all individuals and determines the equilibrium value that balances supply and demand (Nottola et al., 1991, p.191), (de la Maza & Yuret, 1994, p.326). For this reason, neither of these two papers are relevant to the study of equilibration or bargaining behaviors in continuous double auctions.

The apparent lack of work in artificial life on agents with bargaining behaviors for market-based environments is confirmed in a recent review paper by Leigh Tesfatsion, a professor of economics and mathematics at Iowa State University (Tesfatsion, 1997). The paper presents a summary overview of aspects of artificial life especially relevant for the study of decentralized market economies. The two main areas of research activity discussed are: the combination of evolutionary game theory (e.g., iterated prisoner's dilemma) with preferential partner selection (i.e., the ability to choose or refuse particular opponents in the game); and an extension of this, where trade networks can form and evolve. In the studies of trade networks, the prisoner's dilemma game is used to model risky trades between individuals. Thus, there is no emphasis on bargaining mechanisms in this work, and the indications from Tesfatsion (1997) are that little or no work in Artificial Life is comparable to the work on ZIP traders presented here.

6.6 Further Work

While the results presented in the previous section indicate that the ZIP bargaining mechanisms give results more comparable to human traders than do Gode and Sunder's ZI-C traders, there are many possible ways in which this work could be extended.

First, the rationale for the ZIP mechanisms comes from the qualitative arguments of Section 6.1, and while the results are promising, it would be more satisfactory to develop a more rigorous, analytic treatment of these mechanisms. The qualitative rationale could perhaps be supported by game-theory analysis, or the algorithmic complexity could be analysed both in time and in space (e.g. costs for storage and network bandwidth). Furthermore, it would be attractive to develop proofs concerning the convergence to equilibrium of ZIP systems.

Also, the demonstrations of the ZIP traders come from simulations of minimally simple markets, similar to those used in Smith's early experiments. There are a variety of ways in which the complexity of the market environments could be increased, which may reveal the need for revisions or extensions to the basic ZIP mechanisms introduced here.[5]

Relatively straightforward additions to the market structure include endowing the agents with the right to buy or sell multiple units of commodity, with each agent's units having different limit prices, possibly also with the lot-size of each deal being chosen by the agents. A natural next step would then be to have multiple types of commodity, with the possibility of one being a substitute for another.

The use of ZIP mechanisms in agents that have the right to both buy and sell could be investigated, giving the possibility of agents engaging in speculation and arbitrage. Currently, the agents trade in what amounts to a single centralised trading-pit: it would be attractive to explore the dynamics of spatially distributed or segmented markets where agents can only trade with nearby agents: the topology of such distributed markets (e.g. whether a 2-dimensional grid of agents has planar, cylindrical, or toroidal topology) may have a significant effect on dynamics and stability of the markets. Another significant issue to examine is the extent to which the market dynamics are affected by the division of time into discrete 'days': in many applications this assumption may be untenable, and it may be the case that agents 'drop out' of the market and re-enter it in an asynchronous fashion, for varying periods of time. The introduction of delays and noise into distributed markets is also likely to have a significant impact on their dynamics. In particular, delays and noise introduce uncertainty and risk: received signals might be incorrect, through corruption by noise or as a consequence of being out-of-date. Because of

[5]George van Montfort (personal communication, 1997) notes that there is at least one special case where the current ZIP traders will not reach equilibrium. If only sellers can shout, and the initial bid-and-offer arrays give non-intersecting apparent supply and demand curves (because all the buyers' shout prices are lower than all the sellers' limit prices) then all shouts are offers but no shout is ever accepted: adaptation in the sellers will reduce their offer prices, but the buyers will never lower their profit margins and so no transaction will ever occur. It would be necessary to extend the current ZIP trading strategies to solve this problem.

this, it may be necessary for the traders to reason about their 'beliefs' concerning the reliability of the data used in making trading decisions.

As a complement to the current system where ZIP traders buy and sell in a commodity market, ZIP traders could be developed for use in asset markets, where the items bought and sold can generate an income (the dividend stream) while they are owned (see, e.g., Davis and Holt (1993, p.162ff.), Sunder (1995)). Moreover, the current ZIP market is a spot market (transactions are for items bought and sold immediately – "on the spot"): it would be interesting to evaluate the performance of ZIP traders in derivative markets. Derivative markets involve the buying and selling of forward contracts such as futures (i.e., binding obligations) and options (i.e., exercisable rights): each contract is for the purchase or sale of units at a pre-specified price, on or by a given date in the future: for a review of some simple types of derivatives, see Wilmott, Howison, and Dewynne (1995). Derivative markets provide mechanisms for spreading risk and for intertemporal arbitrage.

It is seems very probable that, as market organisations with fewer simplifying constraints are used and as the range of possible actions available to the traders increases, more complex decision and adaptation mechanisms will be required. An obvious first approach would be to introduce "higher-order" adaptation mechanisms so that values which are currently parameters for each agent become variables. That is, the values for the learning rate (β in Equation 13) and momentum (γ in Equation 15) for each agent could be varied dynamically on the basis of that agent's experiences in the market. Also, other variables may be introduced into the adaptation and bargaining mechanisms: there are a number of variables that the current ZIP traders do not take account of which a human trader might use to determine more profitable prices. Examples include: whether there are more buyers than sellers (or more offers than bids shouted) and vice versa; the time remaining until the end of the trading period; predictions of cyclical fluctuations in supply and demand; the average prices of the competition (to allow aggressive or predatory pricing, under-selling or over-bidding to attack the competition); and so on. Also, the current ZIP traders are specified as discrete-time processes, but in more realistic (i.e., more complex) markets, it is likely that continuous-time processes will be required.

Although more traditional machine learning techniques may also be usefully employed, recent work in biologically-inspired computing (so-called "artificial life") has seen the development of a number of adaptation mechanisms that could be employed in automatically adapting or tuning trading agents. Such techniques include the wide variety of neural-network learning algorithms, and evolutionary approaches such as genetic programming and classifier systems. This may allow further exploration or strengthening of the links between evolutionary and economic dynamics (see e.g. Hodgson (1993) and Vromen (1995)). In all cases, the profit accrued by an agent could be used as an obvious reinforcement payoff or reward signal.

6.7 Summary

This section commenced with the simple qualitative arguments, presented in Section 6.1, for how a minimally intelligent trader might operate, and Section 6.2 then discussed some simple quantitative adaptation mechanisms taken from the machine learning literature. Together, these strategy and adaptation define the current ZIP traders.

The results presented in Section 6.3 demonstrated that the ZIP traders yield better results than ZI-C traders: Section 5.3 showed ZI-C traders converging to equilibrium in one market but failing (as predicted) in another three: the ZIP traders do not fail to reach equilibrium in any of these four markets. It was also demonstrated that profit dispersion is lower in ZIP trader markets than in ZI-C markets, so the ZIP results are closer to the human-trader data presented by Gode and Sunder (1993).

The discussion in Section 6.4 compared the results of ZIP traders to Smith's (1962) data from human subjects, in markets where ZI-C traders either fail or cannot be used without extending their specification. The ZIP traders were demonstrated to give results qualitatively similar to those of Smith's human subjects: even the modes of failure are similar in ZIP and human traders. (That is, like humans, ZIP traders showed a slow approach to equilibrium from below in the excess-demand markets of Figures 8, 38, and 40, and a convergence to below-equilibrium prices in the 'retail' market of Figures 9 and 51 to 55.)

Smith also experimented with altering supply and demand mid-way through the experiment, and with 'high-volume' markets where his human subjects were given the right to buy or sell more than one unit per day. Again, ZIP traders exhibit human-like performance in such markets.

These similarities between theoretical predictions, human data, and ZIP traders are striking and significant because of the simplicity of the trading strategies and adaptation mechanisms in the ZIP traders. While Section 5 demonstrated that ZI-C traders are *too* simple, the results in this section indicate that ZIP traders are simple enough to give human-like performance, but not too simple. Having established these baseline results, Section 6.6 sketched out possibilities for extending this work. Clearly, there is much further work that could be done.

7 Conclusion

The development of computational mechanisms that allow groups of software agents to exhibit bargaining behaviors in market-based environments satisfies a number of needs. In market-based control, simple mechanisms are required to give computationally efficient, robust, and truly distributed resource allocation and control in computational ecologies. Such mechanisms could also be employed in the growing field of commerce on the internet. Moreover, such mechanisms act as mechanistically rigorous statements of potential models of human bargaining behaviors, although it is likely that more complex mechanisms would be required to further account for the many subtleties and nuances of human behavior: empirical work in experimental economics and human psychology would also be necessary to validate any models. Once validated, such model agents could be used in the manner intended in the work of Arthur (1993) or Easley and Ledyard (1992), for conveniently testing theories concerning the behavior of humans in different market structures and conditions.

Gode and Sunder's work was an important contribution to the field of experimental economics, providing an absolute lower limit on the mechanistic complexity of trading agents, and demonstrating that allocative efficiency is a poor indicator of the intelligence of agents in a double-auction market. However, the critique presented in Section 5 indicates that some of the tendencies of ZI-C traders towards theoretical equilibrium values are predictable from *a priori* analysis of the statistics of the system. This in turn indicates a need for bargaining mechanisms more complex than the simple stochastic generation of bid and offer prices.

The work on ZIP traders, reported in Section 6, should be viewed as a preliminary sketch of what forms such bargaining mechanisms might take. The ZIP traders are more complex than Gode and Sunder's ZI-C traders, but only slightly, and in any case are manifestly much less complex than humans. Nevertheless, the results from the ZIP traders, both in terms of equilibration and profit dispersion, are clearly closer to those from human experimental markets than are the results from ZI-C traders. It is reassuring to see that very simple mechanisms can give such human-like results, but there is much further work that could be done in exploring behavior of ZIP traders in more complex market environments, and in attempting to extend the behavioral sophistication of such traders without unduly adding to their complexity.

Acknowledgements

This report describes work done while I held a Visiting Academic post at Hewlett-Packard Laboratories Bristol, from February to September 1996. Thanks to Martin Merry, Ray Crispin, and John Taylor for agreeing to have me stay. The early phases of this work were greatly helped by a visit to Palo Alto where I had valuable discussions with Pano Santos at HP Labs, Bernardo Huberman and Tadd Hogg at Xerox PARC, and Mark Miller at Agorics Inc., and by discussions with other members of the then-MBSD group at HP Bristol, including Chris Priest, Susi Ross, Ruud Schoonderwoerd and Erin Stansfield. The main results in this paper were presented at the First Hewlett-Packard International Workshop on Interacting Software Agents, September 1996, and I thank the participants of that meeting for their comments. Finally, most thanks of all to Janet Bruten for her commitment, enthusiasm, support, and involvement in all stages of this work: for suggesting I spend time at HP, taking a risk in agreeing the research topic, sitting through weekly (lack of) progress meetings, tolerating my chaotic schedule, organising the Workshop, arranging equipment loans after I left, and showing Job-like patience in waiting for this report to be finished. However clichéd, it is simply true that none of this would have happened without her.

This page is intentionally blank

A The Code

The system is relatively compact: in total, the source files contain about 2500 lines of code. The code is written in ANSI C, in a fashion that should help porting to C++ or Java: structures and structure-manipulating functions are grouped together where appropriate.

There are seven main source files, most of which have associated header files. Of these, only two (`agent`, discussed in Section A.4, and `smith`, discussed in Section A.8) are directly involved in running the simulations: the others are used to provide data-logging (`ddat` and `tdat` in Sections A.6 and A.5); experiment control (`expctl` in Section A.3); random numbers (`random` in Section A.2); and visualisation of supply and demand curves (`sd` in Section A.7). The Unix system utility `make` may be used with the `makefile` of Section A.9 to compile all the files together: `makefile` also indicates the cross-dependencies between the various header and code files. There is one further header file, used to define system-wide maxima for array bounds, described in Section A.1.

The `makefile` produces an executable called `smith`. The executable requires two arguments: an integer specifying how many experiments should be run, and the name of a control-file. If more than one experiments are to be run, the same experimental conditions are repeated n times (without re-setting the seed of the random number generator), and summary statistics are calculated. The control file specifies a number of important system parameters which determine the experimental conditions, such as the number of buyers, the number of sellers, and their associated supply and demand schedules. Further details of the format for the control file are given in Section A.3.

The `main` function in Section A.8 includes definition of an integer value `verbose` which can be set to zero to suppress output to `stdout` while the experiments are running, or set to positive integer values to allow a running commentary of the action of the system to appear. Verbose is passed from `main` to subordinate functions: setting all calls to `verbose=1` can be useful in debugging or in understanding the system, but can generate megabytes of text.

While the system is running, it generates files in a format suitable for input to the Unix system utility `xgraph`, recording time-series of variables of interest as the trading sessions and days progress. Files illustrating the underlying and apparent supply and demand schedules are also generated in `xfig2.1` format by the routines in `sd` (Section A.7).

Brief notes are provided below to supplement the comments embedded in the code. A sample control-file and the resultant output are provided in Appendix B.

A.1 System Maxima: `max.h`

Rather than use dynamic memory allocation, the code is written to use fixed-length arrays, with the maximum array lengths defined by constants listed in this file. The experiment-control file, read by the routines in `expctl` (Section A.3) allows for virtual array sizes to be specified that are smaller than these system limits. `Max.h` also has system limits on the maximum number of iterations for some of the major loops in the `main` function (Section A.8): again, in several cases these can be overridden by specifying smaller values in the `expctl` control file.

max.h: maxima for array bounds etc

```
#define MAX_N_DAYS 30
#define MAX_TRADES 100
#define TOT_TRADES (MAX_N_DAYS*MAX_TRADES)
#define MAX_FAILS 100  /*maximum numbers of bids/offers allowed to fail before day's trading closes*/

#define MAX_BUYERS 100
#define MAX_SELLERS 100
#define MAX_AGENTS (MAX_BUYERS>MAX_SELLERS?MAX_BUYERS:MAX_SELLERS)
#define MAX_UNITS 3  /*max no. of units an agent can sell/buy*/
#define MAX_SCHED 2  /*max no. of supply or demand schedules in an experiment*/
#define MAX_ID 30    /*max no. of chars in id tag used for output files*/
```

A.2 Random Numbers: `random`

The following routines were used in preference to system-dependent (psuedo-)random number generators. The use of portable, platform-independent random-number routines increases replicability and, in the case of the routines used here, also increases the statistical reliability of the results. The negative side is that these routines are much more computationally expensive than the quick-and-dirty routines typically provided as system calls. The routines in this section are reproduced or adapted from *Numerical Recipes in C: The Art of Scientific Computing* by W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, Cambridge University Press, 1988.

Note also that `random.h` defines the type `Real`, used in place of `float` or `double` to allow easy system-wide switching between different numeric precisions.

A.2.1 random.h

---

random.h
Dave Cliff, May 1991
Some general random routines
Copied or adapted from Numerical recipies in C *by Press, Flannery, Teukolsky, and Vetterling, (CUP, 1988).*

`#define Real double`

`void rseed(int *);` /**reseed random number generator**/

`Real randval(Real);` /**return a (near)uniform distributed random number* $\in [0, limit]$*/

`int irand(int);` /**return a random integer* $\in \{0, \ldots, limit - 1\}$ */

`Real gaussrand(void);` /**returns a* $\mathcal{N}(0,1)$ *random deviate**/

NB: abs(gaussrand()) will be > 3 *about once in 400 trials (the* $3 - \sigma$ *rule).*

`Real exprand(Real);` /**exponential distribution with specified mean**/

A.2.2 random.c

---

random.c
Dave Cliff, May 1991

```
#include <math.h>
#include <time.h>
#include <stdio.h>
#include "random.h"
```

ran1 from the Numerical Recipes in C Book – *it's the slowest but (?) best*
******************* *ran1* *****************

```
#define M1 259200
#define IA1 7141
#define IC1 54773
#define RM1 (1.0/M1)
#define M2 134456
#define IA2 8121
#define IC2 28411
#define RM2 (1.0/M2)
#define M3 243000
#define IA3 4561
#define IC3 51349

float ran1(idum)
int *idum;
{
    static long ix1,ix2,ix3;
    static float r[98];
    float temp;
    static int iff=0;
    int j;
    void nrerror();

    if (*idum < 0 || iff == 0) {
        iff=1;
        ix1=(IC1-(*idum)) % M1;
        ix1=(IA1*ix1+IC1) % M1;
        ix2=ix1 % M2;
        ix1=(IA1*ix1+IC1) % M1;
        ix3=ix1 % M3;
        for (j=1;j<=97;j++) {
            ix1=(IA1*ix1+IC1) % M1;
            ix2=(IA2*ix2+IC2) % M2;
            r[j]=(ix1+ix2*RM2)*RM1;
        }
        *idum=1;
    }
    ix1=(IA1*ix1+IC1) % M1;
    ix2=(IA2*ix2+IC2) % M2;
    ix3=(IA3*ix3+IC3) % M3;
    j=1 + ((97*ix3)/M3);
    if (j > 97 || j < 1)
    /* nrerror("RAN1: This cannot happen."); */
       fprintf(stderr,"RAN1: This cannot happen.");
    temp=r[j];
    r[j]=(ix1+ix2*RM2)*RM1;
    return temp;
}

#undef M1
#undef IA1
#undef IC1
#undef RM1
#undef M2
#undef IA2
#undef IC2
#undef RM2
#undef M3
#undef IA3
#undef IC3
```

****

NB ran1 is not exported – it's masked by the following routines

*rseed: reseed the random number generator from the system clock if (*s)=0 then the system clock is used, otherwise the (*s) is used*

```
void rseed(int *s)
{ time_t tseed;
  int    seed;

  if((*s)==0)
  { time(&tseed);
    seed=(int)(tseed%32767);
    *s=seed; }
  else seed=*s;
  fprintf(stdout,"\n: Seed is %d\n",seed);
  /* srandom(seed); */
  seed=seed*-1;
  ran1(&seed);
}
```

randval: return a (near)uniform distributed random number in the range 0..limit, as a Real

```
Real randval(Real limit)
{ float  rv;
  int    i=1;
  /*get a random value in the range 0..1*/
  rv=ran1(&i);
  return(limit*((Real)rv));
}
```

irand: return a random integer in [0..limit-1]

```
int irand(int limit)
{ int ir;
  /*while loop is used to trap the exceptional case where the underlying deviate in [0,1] actually returns 1.00*/
  ir=limit;
  while(ir==limit)
  { ir=(int)(floor(randval((Real)limit))); }
  return(ir);
}
```

gaussrand: return a $\mathcal{N}(0,1)$ deviate

```
Real gaussrand(void)
{ static int iset=0;
  static Real gset;
  Real fac,r,v1,v2;

  if(iset==0)
  { do { v1=2.0*randval(1.0)-1.0;
         v2=2.0*randval(1.0)-1.0;
         r=(v1*v1)+(v2*v2);
       } while (r>=1.0);
    fac=sqrt(-2.0*log(r)/r);
    gset=v1*fac;
    iset=1;
    return(v2*fac);
  }
  else
  { iset=0;
    return(gset);
  }
}
```

exprand: exponentially distributed variable: also from numerical recipes.

```
Real exprand(Real mean)
{ Real r;

  r=0.0;
  while(r==0.0)
  { r=randval(1.0); }
  return ((-log(r))*mean);
}
```

A.3 Experiment Control: `expctl`

The main parameters of interest in Smith's experiments are read from an "experiment control" data-file, referred to as the *control-file*, provided as a command-line argument to `smith`. The use of control-files allows for batch-processing (e.g. multiple overnight runs); in principle, a graphical user interface could be developed to allow for interactive creation and editing of the control data. Lines in the control-file starting with the character '`#`' are treated as comments, and ignored. Otherwise, control parameters are read from the file in the following order:

Parameter	**Data-type**	**Meaning**
i.d. string	`char[]`	i.d. tag attached to output files, etc.
no. of days	`int` > 0	number of trading 'days'
mintrades	`int` $\in \{1, \ldots, \text{maxtrades}\}$	minimum number of trades per day
maxtrades	`int` $\geq$ mintrades	maximum number of trades per day
random-flag	`int` $\in \{0, 1\}$	flag denoting ZI-C or 'intelligent' traders
NYSE-flag	`int` $\in \{0, 1\}$	flag set to 1 for 'NYSE' trading rule
no. of demand schedules	`int` > 0	number of demand schedules
D-Sched-1	`SD-sched`	first demand schedule
D-Sched-2	`SD-sched`	second demand schedule
...	...	...
D-Sched-n	`SD-sched`	final demand schedule
no. of supply schedules	`int` > 0	number of supply schedules
S-Sched-1	`SD-sched`	first supply schedule
S-Sched-2	`SD-sched`	second supply schedule
...	...	...
S-Sched-m	`SD-sched`	final supply schedule

The meaning of most of the above parameters should be clear to readers with an understanding of Smith's experimental methods. With `NYSE-flag`$= 1$, sellers are only allowed to make offers that improve on (are less than) the current best offer and buyers are only allowed to make bids that improve on (are more than) the current best bid; this auction protocol is used at the New York Stock Exchange (NYSE), among other places. With `NYSE-flag`$= 0$, no such constraint is imposed. In line with Smith's methods, different supply and demand schedules can be introduced during the course of an experiment. The number of demand schedules is specified in the control-file, followed by specifications of each schedule; following this, the number of supply schedules is specified, followed by a specification of each supply schedule. Each schedule should be listed in the control-file in the following order:

Parameter	**Data-type**	**Meaning**
no. of agents	`int`> 0	number of traders
startday	`int`≥ 0	day on which this schedule first applies
endday	`int`$>$startday	last day on which this schedule applies
shout	`int`$\in \{0, 1\}$	flag whether agents can shout or remain passive
agentsched-1	`Agent-sched`	first agent's schedule of units
agentsched-2	`Agent-sched`	second agent's schedule of units
...	...	...
agentsched-n	`Agent-sched`	final agent's schedule of units

Where the `Agent-sched` data structure is simply a specification of the number of units that agent is given to buy or sell, and the limit price for each unit; specified in the control-file as:

Parameter	**Data-type**	**Meaning**
no. of units	`int`> 0	how many units the agent has to buy or sell
unit-price-1	`Real`> 0	limit price of first unit
unit-price-2	`Real`> 0	limit price of second unit
...	...	...
unit-price-p	`Real`> 0	limit price of last unit

The header file `expctl.h` contains the structure definitions for `Agent_sched` and `SD_sched`, along with a structure `Expctl` which holds all the control parameter values once they have been read from the file and (partially) validated by the function `expctl_in` defined in the file `expctl.c`. An example control file is shown in Section B.1.

A.3.1 expctl.h

expctl.h: header file for experiment control data struct and i/o etc
Dave Cliff
Aug 1996

Agent-sched: data associated with one agent's buy/sell limits etc

```
typedef struct an_agent_sched{
  int n_units;           /*how many units the agent has/wants*/
  Real limit[MAX_UNITS]; /*limit price of each unit*/
} Agent_sched;
```

SD-sched: data associated with a supply or demand schedule

```
typedef struct sd_sched{
  int n_agents;                    /*how many agents involved*/
  int first_day;                   /*first day this schedule applies to*/
  int last_day;                    /*last day this schedule applies to*/
  int can_shout;                   /*boolean: 0=¿silent traders; 1=¿can shout*/
  Agent_sched agents[MAX_AGENTS]; /*details of individual agents*/
} SD_sched;
```

Expctl: experiment control parameters

```
typedef struct a_expctl{
  char id[MAX_ID];                /*id characters for output files*/
  int n_days;                     /*number of trading periods to run for*/
  int min_trades;                 /*minimum number of trades per day*/
  int max_trades;                 /*maximum number of trades per day*/
  int random;                     /*boolean: 0=¿ ZIP; 1=¿ZI-C*/
  int nyse;                       /*boolean: 0=¿NYSE off; 1=¿NYSE on*/
  int n_dem_sched;                /*number of demand schedules*/
  SD_sched dem_sched[MAX_SCHED]; /*details of demand schedules*/
  int d_sched;                    /*index of currently active demand schedule*/
  int n_sup_sched;                /*number of supply schedules*/
  SD_sched sup_sched[MAX_SCHED]; /*details of supply schedules*/
  int s_sched;                    /*index of currently active supply schedule*/
} Expctl;

void expctl_in(char [],Expctl *,int);
```

A.3.2 expctl.c

---

expctl.c: read experiment control parameters from a file
Dave Cliff
Aug 1996

This does some *validity checks but still need to be careful that the data-file it reads from is structured correctly. When there is more than one schedule for supply or demand, they must be listed in the data-file in the order they are to become active. The first-day for the 0th schedule is set to zero, whatever value is given in the data-file*

```
#include <math.h>
#include <stdio.h>
#include <ctype.h>

#include "random.h"
#include "max.h"
#include "expctl.h"

#define LLEN 1024 /*max. no. of characters in a line*/
```

get-non-comment-line: read to start of next line that doesn't start with '#'

```
int get_non_comment_line(FILE *fp)
{ int c,reading=1;
  char s[LLEN];

  while(reading)
  { /*get to first non-whitespace char*/
    c=' ';
    while(isspace(c))
    { c=fgetc(fp);
      if(c==EOF) return(EOF);
    }

    /*is this a comment line?*/
    if(c=='#')
    { /*yes: read the rest of this line*/
      ungetc(c,fp);
      fgets(s,LLEN,fp);
    }
    else
    { /*no: put the char back and exit*/
      ungetc(c,fp);
      return(EOF-1); /*i.e. something that isn't EOF*/
    }
  }
}
```

read-sched: read a supply or demand schedule

```
int read_sched(FILE *fp,SD_sched *sched,int verbose)
{ int i,*pi,a,u;
  float f,*pf;

  pi=&i; pf=&f;
```

```
/*read number of agents*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { sched->n_agents=(*pi);
    if((sched->n_agents<1)||(sched->n_agents>MAX_AGENTS))
    { fprintf(stderr,"\nFail: # agents must be in range {1,...,%d}\n",
              MAX_AGENTS);
      exit(0);
    }
    if(verbose)
    { fprintf(stdout,"  %d agents: ",sched->n_agents); fflush(stdout); }
  }
  else {fprintf(stderr,"\nFail: can't read # agents \n"); exit(0);}
}
else {fprintf(stderr,"\nFail: EOF reading # agents\n"); exit(0);}

/*read start day*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { sched->first_day=(*pi);
    if(verbose)
    { fprintf(stdout,"from day %d ",sched->first_day); fflush(stdout); }
  }
}

/*read end day*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { sched->last_day=(*pi);
    if(sched->last_day<sched->first_day)
    { fprintf(stderr,"\nFail: last_day(%d)<first_day(%d)\n",
              sched->last_day,sched->first_day);
      exit(0);
    }
    if(verbose)
    { fprintf(stdout,"to day %d\n",sched->last_day); fflush(stdout); }
  }
}

/*read shout flag*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { sched->can_shout=(*pi);
    if((sched->can_shout<0)||(sched->can_shout>1))
    { fprintf(stderr,"\nFail: can_shout not Boolean (%d)\n",
              sched->can_shout);
      exit(0);
    }
    if(verbose)
    { if(sched->can_shout)
      { fprintf(stdout,"(These traders CAN SHOUT)\n"); fflush(stdout); }
      else
      { fprintf(stdout,"(These traders are SILENT)\n"); fflush(stdout); }
    }
  }
}
```

```
  /*read agent pricing specs*/
  for(a=0;a<sched->n_agents;a++)
  { if(get_non_comment_line(fp)!=EOF)
    { if(fscanf(fp,"%d",pi)!=EOF)
      { sched->agents[a].n_units=(*pi);
        if((sched->agents[a].n_units<1)||(sched->agents[a].n_units>MAX_UNITS))
        { fprintf(stderr,"\nFail: # units must be inrange {1,...,%d}\n",
                   MAX_UNITS);
          exit(0);
        }

        if(verbose)
        { fprintf(stdout,"  Agent %2d, %d units: ",a,sched->agents[a].n_units);
          fflush(stdout);
        }

        for(u=0;u<sched->agents[a].n_units;u++)
        { if(fscanf(fp,"%f",pf)!=EOF)
          { sched->agents[a].limit[u]=(Real)(*pf);
            if(sched->agents[a].limit[u]<0.0)
            { fprintf(stderr,"\nFail: negative price (%f)\n",*pf);
              exit(0);
            }
            if(verbose)
            { fprintf(stdout,"%f ",sched->agents[a].limit[u]);
              fflush(stdout);
            }
          }
        }

        if(verbose) {fprintf(stdout,"\n"); fflush(stdout); }
      }
    }
  } /*end of reading the agent data*/
}
```

expctl-in: read expctl data from a specified file

```
void expctl_in(char filename[],Expctl *ec,int verbose)
{ int *pi,i,sched;
  float f,*pf;
  FILE *fp;

  fp=fopen(filename,"r");
  if(fp==NULL)
  { fprintf(stderr,"\nFAIL: can't open \"%s\" as expctl input file\n",filename);
    exit(0);
  }

  pi=&i; pf=&f;

  /*read id string*/
  if(get_non_comment_line(fp)!=EOF)
  { /*copy id string up to but not including the newline*/
    fscanf(fp,"%s\n",&(ec->id));
    if(verbose)
    { fprintf(stdout,"ID: %s\n",ec->id); fflush(stdout); }
  }
```

```
/*read number of days*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { ec->n_days=(*pi);
    if((ec->n_days<1)||(ec->n_days>MAX_N_DAYS))
    { fprintf(stderr,"\nFail: # trading days must be in range {1,...,%d}\n",
              MAX_N_DAYS);
      exit(0);
    }
    if(verbose)
    { fprintf(stdout,"%d days: ",ec->n_days); fflush(stdout); }
  }
  else { fprintf(stderr,"\nFail: can't read number of days\n"); exit(0); }
}
else { fprintf(stderr,"\nFail: EOF reading number of days\n"); exit(0); }

/*read min number of trades per day*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { ec->min_trades=(*pi);
    if((ec->min_trades<1)||(ec->min_trades>MAX_TRADES))
    { fprintf(stderr,"\nFail: min # trades must be in range {1,...,%d}\n",
              MAX_TRADES);
      exit(0);
    }
    if(verbose)
    { fprintf(stdout,"min_trades=%d ",ec->min_trades); fflush(stdout); }
  }
  else { fprintf(stderr,"\nFail: can't read min_trades\n"); exit(0); }
}
else { fprintf(stderr,"\nFail: EOF reading min_trades\n"); exit(0); }

/*read max number of trades per day*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { ec->max_trades=(*pi);
    if((ec->max_trades<ec->min_trades)||(ec->max_trades>MAX_TRADES))
    { fprintf(stderr,"\nFail: max # trades muts be in range {%d,...,%d}\n",
              ec->min_trades,MAX_TRADES);
      exit(0);
    }
    if(verbose)
    { fprintf(stdout,"max_trades=%d\n",ec->max_trades); fflush(stdout); }
  }
  else { fprintf(stderr,"\nFail: can't read max_trades\n"); exit(0); }
}
else { fprintf(stderr,"\nFail: EOF reading max_trades\n"); exit(0); }

/*read random flag*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { ec->random=(*pi);
    switch(ec->random)
    { case 1: if(verbose) fprintf(stdout,"Random (ZI-C) traders; ");
              break;

      case 0: if(verbose) fprintf(stdout,"Intelligent traders; ");
              break;
      default: fprintf(stderr,"\nFail: random flag must be boolean\n");
               exit(0);
    }
    if(verbose) fflush(stdout);
  }
  else { fprintf(stderr,"\nFail: can't read random flag\n"); exit(0); }
}
else { fprintf(stderr,"\nFail: EOF reading random flag\n"); exit(0); }

/*read nyse flag*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { ec->nyse=(*pi);
    switch(ec->nyse)
    { case 1: if(verbose) fprintf(stdout,"NYSE trading rules\n");
              break;

      case 0: if(verbose) fprintf(stdout,"no NYSE rules\n");
              break;

      default: fprintf(stderr,"\nFail: NYSE flag must be boolean\n");
               exit(0);
    }
    if(verbose) fflush(stdout);
  }
  else { fprintf(stderr,"\nFail: can't read nyse flag\n"); exit(0); }
}
else { fprintf(stderr,"\nFail: EOF reading nyse flag\n"); exit(0); }

/*read number of demand schedules*/
if(get_non_comment_line(fp)!=EOF)
{ if(fscanf(fp,"%d",pi)!=EOF)
  { ec->n_dem_sched=(*pi);
    if((ec->n_dem_sched<1)||(ec->n_dem_sched>MAX_SCHED))
    { fprintf(stderr,"\nFail: # demand scheds must be in range {1,...,%d}\n",
              MAX_SCHED);
      exit(0);
    }

    if(verbose)
    { fprintf(stdout,"%d demand schedules:\n",ec->n_dem_sched);
      fflush(stdout);
    }
  }

  else {fprintf(stderr,"\nFail: can't read # demand schedules\n"); exit(0);}
}
else {fprintf(stderr,"\nFail: EOF reading # demand schedules\n"); exit(0);}

/*read the schedules*/
for(sched=0;sched<ec->n_dem_sched;sched++)
{ if(verbose) fprintf(stdout,"  Demand schedule %d:\n",sched);
  if(read_sched(fp,&(ec->dem_sched[sched]),verbose)==EOF)
  { fprintf(stderr,"\nFail: no more demand schedules\n");
    exit(0);
  }
}

ec->d_sched=0;
ec->dem_sched[ec->d_sched].first_day=0;
```

```
  /*read number of supply schedules*/
  if(get_non_comment_line(fp)!=EOF)
  { if(fscanf(fp,"%d",pi)!=EOF)
    { ec->n_sup_sched=(*pi);
      if((ec->n_sup_sched<1)||(ec->n_sup_sched>MAX_SCHED))
      { fprintf(stderr,"\nFail: # supply scheds must be in range {1,...,%d}\n",
                 MAX_SCHED);
        exit(0);
      }

      if(verbose)
      { fprintf(stdout,"%d supply schedules:\n",ec->n_sup_sched);
        fflush(stdout);
      }
    }
    else {fprintf(stderr,"\nFail: can't read # supply schedules\n"); exit(0);}
  }
  else {fprintf(stderr,"\nFail: EOF reading # supply schedules\n"); exit(0);}

  /*read the schedules*/
  for(sched=0;sched<ec->n_sup_sched;sched++)
  { if(verbose) fprintf(stdout,"  Supply schedule %d:\n",sched);
    if(read_sched(fp,&(ec->sup_sched[sched]),verbose)==EOF)
    { fprintf(stderr,"\nFail: no more supply schedules\n");
      exit(0);
    }
  }

  ec->s_sched=0;
  ec->sup_sched[ec->s_sched].first_day=0;

  fclose(fp);
}
```

A.4 The Agent: `agent`

The definition for the data structure `Agent` associated with each trading agent is given in `agent.h`. The functions in `agent.c` allow agents to be initialised, and have their dealing strategies altered in response to the ongoing stream of offers and bids accepted and declined.

A.4.1 agent.h

---

agent.h: general global constants and structures
Dave Cliff
August 1996

```
#define NULL_EQ -1 /*signals no equilibrium*/
```

symbolic constants for agent type, shout type, and whether shout is accepted or rejected

```
#define BUY 1
#define SELL 0
#define BID 1
#define OFFER 0
#define DEAL 1
#define NO_DEAL 0
#define END_DAY 2

typedef struct  an_agent{
    int     job;     /*BUYing or SELLing*/
    int     active;  /*still in the market?*/
    int     n;       /*number of deals done*/
    int     willing; /*want to make a trade at this price?*/
    int     able;    /*allowed to trade at this price?*/
    Real    limit;   /*the bottom-line price for this agent*/
    Real    profit;  /*profit coefficient in determinining bid/offer price*/
    Real    beta;    /*coeff for changing profit over time (learning rate)*/
    Real    momntm;  /*momentum in changing profit*/
    Real    last_d;  /*last change*/
    Real    price;   /*what the agent will actually bid*/
    Real    quant;   /*how much of this commodity*/
    Real    bank;    /*how much money this agent has in the bank*/
    Real    a_gain;  /*actual gain*/
    Real    t_gain;  /*theoretical gain*/
    Real    sum;     /*in determining average reward*/
    Real    avg;     /*average reward*/
} Agent;

void set_price(Agent *);
void shout_update(int deal_type,int status,
                 int n_sell,Agent sellers[],int n_buy,Agent buyers[],Real price,
                 int verbose);
void buy_init(Agent b[],int verbose);
void sell_init(Agent s[],int verbose);
int willing_trade(Agent *a,Real price);
void profit_alter(Agent *a,Real price,int verbose);
```

A.4.2 agent.c

---

agent.c: defines an agent, how it adapts, etc.
Dave Cliff
Aug 1996

```
#include <math.h>
#include <stdio.h>
#include "random.h"
#include "max.h"
#include "agent.h"

#define BONUS 0.00

#define MARKUP 1.1
#define MARKDOWN 0.9
#define MARK 0.05
```

set-price: set the price of an agent from its limit and profit values

```
void set_price(Agent *a)
{ a->price=(a->limit)*(1+a->profit);
  /*normalise to one-cent precision*/
  a->price=(floor((a->price*100)+0.5))/100;
}
```

agent-init: initialise the common elements of an agent (buyer or seller)

```
void agent_init(Agent *a,int verbose)
{
  a->beta=0.1+randval(0.4);
  a->bank=0.0;
  a->n=0;
  a->sum=0.0;
  a->last_d=0.0;
  a->momntm=0.2+randval(0.6);
  a->momntm=randval(0.1);
  a->active=1;
  if(verbose)
  { fprintf(stdout,"prof=%+5.3f beta=%5.3f mom=%5.3f bank=%5.2f\n",
            a->profit,a->beta,a->momntm,a->bank);
  }
}
```

buy-init: initialize the buyers

```
void buy_init(Agent b[MAX_AGENTS],int verbose)
{ int a;

  for(a=0;a<MAX_AGENTS;a++)
  { b[a].job=BUY;
    b[a].profit=-1.0*(0.05+randval(0.3));
    if(verbose) fprintf(stdout,"B%2d ",a);
    agent_init(b+a,verbose);
  }
}
```

sell-init: initialize the sellers

```
void sell_init(Agent s[MAX_AGENTS],int verbose)
{ int a;

  for(a=0;a<MAX_AGENTS;a++)
  { s[a].job=SELL;
    s[a].profit=0.05+randval(0.3);
    if(verbose) fprintf(stdout,"S%2d ",a);
    agent_init(s+a,verbose);
  }
}
```

willing-trade: is an agent willing to trade at given price?

```
int willing_trade(Agent *a,Real price)
{ if(a->job==BUY)
  { /*willing to buy at this price?*/
    if((a->active)&&(a->price>=price))
    { a->willing=1; }
    else
    { a->willing=0; }
  }

  else
  { /*willing to sell at this price?*/
    if((a->active)&&(a->price<=price))
    { a->willing=1; }
    else
    { a->willing=0; }
  }
  return(a->willing);
}
```

profit-alter: update profit margin on basis of sale price using Widrow-Hoff style update with learning rate β.

```
void profit_alter(Agent *a,Real price,int verbose)
{ Real c,diff,change,newprofit;

  if(verbose) fprintf(stdout,"lim=%5.3f prof=%5.3f price=%5.2f",
                      a->limit,a->profit,a->price);

    diff=(price-(a->price));
    change=((1.0-(a->momntm))*(a->beta)*diff)+((a->momntm)*(a->last_d));

    if(verbose) fprintf(stdout," last_d=%5.3f diff=%5.2f chng=%+5.3f",
                               a->last_d,diff,change);

    a->last_d=change;

    /*set new prices by altering profit margin*/
    newprofit=((a->price+change)/a->limit)-1.0;

  if(a->job==SELL)
  { if(newprofit>0.0) a->profit=newprofit; }
  else
  { if(newprofit<0.0) a->profit=newprofit; }

  set_price(a);
  if(verbose)
  { fprintf(stdout," nu_prof=%5.3f nu_price=%5.2f",a->profit,a->price);}
}
```

shout-update: update strategies of buyers and sellers after a shout

```
void shout_update(int deal_type,int status,int n_sell,
                  Agent sellers[],int n_buy,Agent buyers[],Real price,
                  int verbose)
{ int b,s;
  Real target_price;
  /*any seller whose price is less than or equal to the deal price raises profit margin*/
  /*(this is an attempt to increase profits next time around)*/

  for(s=0;s<n_sell;s++)
  { if(verbose) fprintf(stdout,"S%02d(%d) ",s,sellers[s].active);

    if(status==DEAL)
    { if(sellers[s].price<=price)
      { /*could get more? - try raising margin*/
        target_price=(price*(1.0+randval(MARK)))+randval(0.05);
        profit_alter(sellers+s,target_price,verbose);
      }

      else
      { /*wouldn't have got this deal, so mark the price down*/
        if( (deal_type==BID) &&
            (!willing_trade(sellers+s,price)) &&
            (sellers[s].active)
          )
        { target_price=(price*(1.0-randval(MARK)))-randval(0.05);
          profit_alter(sellers+s,target_price,verbose);
        }
      }
    }

    else /*NO DEAL*/
    { if(deal_type==OFFER)
      if((sellers[s].price>=price)&&(sellers[s].active))
      { /*would have asked for more and lost the deal, so reduce profit*/
        target_price=(price*(1.0-randval(MARK)))-randval(0.05);
        profit_alter(sellers+s,target_price,verbose);
      }
    }
    if(verbose)fprintf(stdout,"\n");
  }

  for(b=0;b<n_buy;b++)
  { if(verbose) fprintf(stdout,"B%02d(%d) ",b,buyers[b].active);

    if(status==DEAL)
    { if(buyers[b].price>=price)
      { /*could get lower price? - try raising margin (i.e. cutting price)*/
        target_price=(price*(1.0-randval(MARK)))-randval(0.05);
        profit_alter(buyers+b,target_price,verbose);
      }
      else
      { /*wouldn't have got this deal, so mark the price up (reduce profit)*/
        if( (deal_type==OFFER) &&
            (!willing_trade(buyers+b,price)) &&
            (buyers[b].active)
          )
        { target_price=(price*(1.0+randval(MARK)))+randval(0.05);
          profit_alter(buyers+b,target_price,verbose);
        }
      }
    }

    else /*NO-DEAL*/
    { if(deal_type==BID)
      if((buyers[b].price<=price)&&(buyers[b].active))
      { /*would have bid less and also lost the deal, so reduce profit*/
        target_price=(price*(1.0+randval(MARK)))+randval(0.05);
        profit_alter(buyers+b,target_price,verbose);
      }
    }
    if(verbose)fprintf(stdout,"\n");
  }
}
```

A.5 Trading Data: `tdat`

The file `tdat.h` defines a structure for recording the details of a single trade: the transaction price, whether the shout was a bid or offer, and what the theoretical and actual equilibrium price and quantities were at the time of the trade. The file `tdat.c` then defines a function which writes a file of the time series of these data, in a format suitable for display by the Unix system utility `xgraph`.

A.5.1 tdat.h

---

tdat.h: header for routines that record and display data in one trading session
Dave Cliff
Aug 1996

```
typedef struct trade_data{
  Real deal_p; /*price at which deal succeeds*/
  int  deal_t; /*type of deal accepted (bid or ask)*/
  Real t_eq_p; /*theoretical equilibrium price*/
  int  t_eq_q; /*theoretical equilibrium quantity*/
  Real a_eq_p; /*actual equilibrium price*/
  int  a_eq_q;  /*actual equilibrium quantity*/
} Trade_data;

void xg_trades_graph(Trade_data tdat[][MAX_TRADES],
                     Day_data*,int,int,char [],int);
```

A.5.2 tdat.c

---

tdat.c: routine for manipulating data and stats recorded during a single trade, i.e., the auction leading to one deal
Dave Cliff
Aug 1996

```
#include <stdio.h>
#include <math.h>

#include "random.h"
#include "agent.h"
#include "max.h"
#include "ddat.h"
#include "tdat.h"
```

xg-trades-graph: plot stats concerning individual trades in `xgraph` *format*

```
void xg_trades_graph(Trade_data tdat[MAX_N_DAYS][MAX_TRADES],
                     Day_data *ddat,
                     int n_days,int max_trades,
                     char filename[],int n_exps)
{ Real gx,dgx,q;
  int t,d;
  FILE *fp;
  fp=fopen(filename,"w");

  fprintf(fp,"TitleText: %s: n=%d\n\n",filename,n_exps);

  dgx=(1.0/((Real)(max_trades)));

  fprintf(fp,"\"Price\n");
  for(d=0;d<n_days;d++)
  { gx=d+1;
    q=(ddat+d)->quant.sum;
    for(t=0;t<q;t++)
    { if(tdat[d][t].deal_p>=0.0) fprintf(fp,"%f %f\n",gx,tdat[d][t].deal_p);
      gx+=dgx;
    }
  }

  fprintf(fp,"\n\"Actual EqP\n");
  for(d=0;d<n_days;d++)
  { gx=d+1;
    q=(ddat+d)->quant.sum;
    for(t=0;t<q;t++)
    { if(tdat[d][t].a_eq_q!=NULL_EQ) fprintf(fp,"%f %f\n",gx,tdat[d][t].a_eq_p);
      gx+=dgx;
    }
  }

  fprintf(fp,"\n\"Theoretical EqP\n");
  for(d=0;d<n_days;d++)
  { gx=d+1;
    q=(ddat+d)->quant.sum;
    for(t=0;t<q;t++)
    { if(tdat[d][t].t_eq_q!=NULL_EQ) fprintf(fp,"%f %f\n",gx,tdat[d][t].t_eq_p);
      gx+=dgx;
    }
  }

  fclose(fp);
}
```

A.6 Day Data: `ddat`

In one 'day' there will generally be more than one transaction. In both Smith's experiments and those of Gode and Sunder, a variety of statistics are calculated at the end of a day, using the data for the trades that have occurred in that day; plots of time series of these daily data may then be produced. The file `ddat.h` contains structure declarations for recording day data: both single time-series and also aggregate time-series where the mean and standard deviation (s.d.) are of interest.

A.6.1 ddat.h

---

ddat.h: header for ddat.c routines
Dave Cliff
Sept 1996

datatype for Real sum and sum of squares, used in calculuating mean and s.d.

```
typedef struct real_stat{
  Real sum,sumsq;
  int n;
} Real_stat;
```

data and stats for a day's trading

```
typedef struct day_data{
  Real_stat alpha; /*Smith's alpha*/
  Real_stat quant; /*Quantity*/
  Real_stat effic; /*Efficiency*/
  Real_stat price; /*price*/
  Real_stat pdisp; /*profit dispersal*/
  Real_stat volty; /*transaction price volatility*/
} Day_data;
```

ddat-init: initialise daily data

```
void ddat_init(Day_data *);
```

ddat-update: update daily data

```
void ddat_update(Day_data *,int,Real,Real,Real,Real,Real);
```

ddat-xgraph: plot the daily stats in xgraph format

```
void xg_daily_graph(Day_data dd[],int,int,char *);
```

A.6.2 ddat.c

---

ddat.c: code for handling data/stats compiled at end of each trading day
Dave Cliff
Sept 1996

```
#include <math.h>
#include <stdio.h>
#include <string.h>
```

```
#include "random.h"
#include "ddat.h"
#include "max.h"

#define SMALLREAL 0.0000001 /*used to dodge rounding errors on sqrt */

#define DD_ALPHA 0
#define DD_QUANT 1
#define DD_EFFIC 2
#define DD_PRICE 3
#define DD_PDISP 4
#define DD_VOLTY 5
```

rstat-zero: set everything to zero in one Real-stat structure

```
void rstat_zero(Real_stat *r)
{ r->sum=0.0;
  r->sumsq=0.0;
  r->n=0;
}
```

ddat-init: initialise day data

```
void ddat_init(Day_data *ddat)
{
  rstat_zero(&(ddat->alpha));
  rstat_zero(&(ddat->quant));
  rstat_zero(&(ddat->effic));
  rstat_zero(&(ddat->price));
  rstat_zero(&(ddat->pdisp));
  rstat_zero(&(ddat->volty));
}
```

ddat-update: update day data.

```
void ddat_update(Day_data *dd,int n_deals,
                 Real sum_price,Real alpha,Real pdisp,Real effic,Real pdiff)
{ Real v;
  if(n_deals>0)
  { (dd->price.sum)+=(sum_price/n_deals);
    (dd->price.sumsq)+=((sum_price/n_deals)*(sum_price/n_deals));
    (dd->price.n)++;

     v=sqrt(pdiff/n_deals); /*root mean square difference*/
    (dd->volty.sum)+=v;
    (dd->volty.sumsq)+=(v*v);
    (dd->volty.n)++;

    (dd->alpha.sum)+=alpha;
    (dd->alpha.sumsq)+=(alpha*alpha);
    (dd->alpha.n)++;

    (dd->effic.sum)+=effic;
    (dd->effic.sumsq)+=(effic*effic);
    (dd->effic.n)++;

    (dd->quant.sum)+=n_deals;
    (dd->quant.sumsq)+=(n_deals*n_deals);
    (dd->quant.n)++;
    (dd->pdisp.sum)+=pdisp;
    (dd->pdisp.sumsq)+=(pdisp*pdisp);
    (dd->pdisp.n)++;
  }
}
```

ddat-meanpmsd: plot mean plus and minus one standard deviation

```
void ddat_meanpmsd(FILE *fp,int field,int n_days,int n_exps,Day_data dd[])
{ int d,n[MAX_N_DAYS];
  Real mean,meansq,diff,sum[MAX_N_DAYS],sumsq[MAX_N_DAYS];
  char fieldstr[30];

  if(n_days>MAX_N_DAYS)
  { fprintf(stderr,"\nFAIL: MAX_N_DAYS too small in ddat.c: recompile\n");
    exit(0);
  }

  switch(field)
  { case DD_ALPHA: strcpy(fieldstr,"Alpha");
                   for(d=0;d<n_days;d++)
                   { sum[d]=dd[d].alpha.sum;
                     sumsq[d]=dd[d].alpha.sumsq;
                     n[d]=dd[d].alpha.n;
                   }
                   break;

    case DD_QUANT: strcpy(fieldstr,"Quantity");
                   for(d=0;d<n_days;d++)
                   { sum[d]=dd[d].quant.sum;
                     sumsq[d]=dd[d].quant.sumsq;
                     n[d]=dd[d].quant.n;
                   }
                   break;

    case DD_EFFIC: strcpy(fieldstr,"Efficiency");
                   for(d=0;d<n_days;d++)
                   { sum[d]=dd[d].effic.sum;
                     sumsq[d]=dd[d].effic.sumsq;
                     n[d]=dd[d].effic.n;
                   }
                   break;

    case DD_PRICE: strcpy(fieldstr,"Price");
                   for(d=0;d<n_days;d++)
                   { sum[d]=dd[d].price.sum;
                     sumsq[d]=dd[d].price.sumsq;
                     n[d]=dd[d].price.n;
                   }
                   break;

    case DD_PDISP: strcpy(fieldstr,"Dispersion");
                   for(d=0;d<n_days;d++)
                   { sum[d]=dd[d].pdisp.sum;
                     sumsq[d]=dd[d].pdisp.sumsq;
                     n[d]=dd[d].pdisp.n;
                   }
                   break;
    case DD_VOLTY: strcpy(fieldstr,"Volatility");
                   for(d=0;d<n_days;d++)
                   { sum[d]=dd[d].volty.sum;
                     sumsq[d]=dd[d].volty.sumsq;
                     n[d]=dd[d].volty.n;
                   }
                   break;

    default: fprintf(stderr,"\nFAIL: bad field in ddat_meanpmsd (%d)\n",field);
             exit(0);
  }

  fprintf(fp,"\" %s (mean)\n",fieldstr);
  for(d=0;d<n_days;d++) fprintf(fp,"%d %f\n",d+1,sum[d]/n[d]);
  fprintf(fp,"\n");

  fprintf(fp,"\" %s (-1s.d.)\n",fieldstr);
  for(d=0;d<n_days;d++)
  { mean=sum[d]/n[d];
    meansq=mean*mean;
    diff=(sumsq[d]/n[d])-meansq;
    if(diff<SMALLREAL) diff=0.0;
    fprintf(fp,"%d %f\n",d+1,mean-sqrt(diff));
  }
  fprintf(fp,"\n");

  fprintf(fp,"\" %s (+1s.d.)\n",fieldstr);
  for(d=0;d<n_days;d++)
  { mean=sum[d]/n[d];
    meansq=mean*mean;
    diff=(sumsq[d]/n[d])-meansq;
    if(diff<SMALLREAL) diff=0.0;
    fprintf(fp,"%d %f\n",d+1,mean+sqrt(diff));
  }
  fprintf(fp,"\n");
}
```

ddat-xgraph: plot the daily stats in xgraph format

```
void xg_daily_graph(Day_data dd[],int n_days,int n_exps,char *fname)
{ int d;
  FILE *fp;

  fp=fopen(fname,"w");

  fprintf(fp,"TitleText: %s: n=%d\n\n",fname,n_exps);

  if(n_exps<2)
  { /*no sense in calculating SD*/

    fprintf(fp,"\" Alpha\n");
    for(d=0;d<n_days;d++) fprintf(fp,"%d %f\n",d+1,dd[d].alpha.sum);
    fprintf(fp,"\n");

    fprintf(fp,"\" Efficiency\n");
    for(d=0;d<n_days;d++) fprintf(fp,"%d %f\n",d+1,dd[d].effic.sum);
    fprintf(fp,"\n");
    fprintf(fp,"\" Quantity\n");
    for(d=0;d<n_days;d++) fprintf(fp,"%d %f\n",d+1,dd[d].quant.sum);
    fprintf(fp,"\n");

    fprintf(fp,"\" Dispersion\n");
    for(d=0;d<n_days;d++) fprintf(fp,"%d %f\n",d+1,dd[d].pdisp.sum);
    fprintf(fp,"\n");
  }

  else
  { /*plot mean and s.d. for the daily stats*/
    ddat_meanpmsd(fp,DD_PRICE,n_days,n_exps,dd);
    ddat_meanpmsd(fp,DD_ALPHA,n_days,n_exps,dd);
    ddat_meanpmsd(fp,DD_EFFIC,n_days,n_exps,dd);
    ddat_meanpmsd(fp,DD_QUANT,n_days,n_exps,dd);
    ddat_meanpmsd(fp,DD_PDISP,n_days,n_exps,dd);
    ddat_meanpmsd(fp,DD_VOLTY,n_days,n_exps,dd);
  }
  fclose(fp);
}
```

A.7 Supply and Demand: `sd`

Visualisation of the supply and demand curves is of great use in understanding the dynamics of the system: as sellers and buyers make deals, they may be required to withdraw from the market, and their absence can shift both the underlying supply and demand curves (calculated on the basis of the limit prices of the traders). The apparent supply and demand curves, calculated on the basis of the prices the traders are 'shouting' in the auction, can undergo variations both when traders leave the market, and when traders adjust their shout-prices on the basis of their observations of other shouts in the market.

The files `sd.h` and `sd.c` define a function `supdem` which will return either the underlying (theoretical) or apparent (actual) values of equilibrium price, equilibrium quantity, and maximum available surplus. If a non-null character string is given in the argument `fname`, a file is produced which can be displayed using the Unix system utility `xfig`, version 2.1. The functions with names commencing `xf_` defined in `sd.c` exist to mask the intricacies of the `xfig2.1` file format. Full details of the `xfig2.1` file format are available from:

`http://www.cs.virginia.edu/helpnet/Authoring_Tools/xfig/FORMAT2.1.html`

A sample `supdem` output file is given in Section B.3.

A.7.1 sd.h

---

supdem.h
Dave Cliff
Sept 1996

acouple of symbolic constants

```
#define EQ_THEORY 0
#define EQ_ACTUAL 1

void supdem(int,Agent *,int,Agent *,int,Real *,int *,Real *,int,
            char [],Real *,int);
```

A.7.2 sd.c

---

sd.c – code for working with stepped supply and demand curves, producing output in xfig format
Dave Cliff
Sept 1996

```
#include <math.h>
#include <stdio.h>
#include "random.h"
#include "agent.h"
#include "max.h"
#include "sd.h"

#define MAX_PRICES ((MAX_AGENTS)*MAX_UNITS)
```

max number of points in a polyline

```
#define MAX_POINTS (MAX_PRICES*3)
```

constants for Xfig drawing

```
#define SCALE_FACTOR 1200 /*pixels per inch*/

#define X_INCHES 7
#define Y_INCHES 6
#define LMARGIN_X ((int)(0.15*X_INCHES*SCALE_FACTOR))
#define LMARGIN_Y ((int)(0.15*Y_INCHES*SCALE_FACTOR))
#define GRAPH_X   ((int)(0.70*X_INCHES*SCALE_FACTOR))
#define GRAPH_Y   ((int)(0.70*Y_INCHES*SCALE_FACTOR))
#define Y_EQ_0    LMARGIN_Y+GRAPH_Y
#define X_EQ_0    LMARGIN_X+GRAPH_X
#define AX_THICK 1                     /*thickness of axis lines*/
#define AX_PTS  18                     /*pointsize for labelling axes*/
#define TICK_X  ((int)(0.025*GRAPH_Y)) /*x tick height*/
#define TICK_Y  ((int)(0.025*GRAPH_X)) /*y tick length*/
#define X_TICKS 10 /*aim at this number of ticks on the x axis*/
#define Y_TICKS 10 /*aim at this number of ticks on the y axis*/

#define MAX_LABELLEN 80 /*maximum number of characters in a label*/
#define PL_SOLID 0    /*polyline solid linestyle*/
#define PL_DASHED 1   /*polyline dashed linestyle*/
#define PL_DOTTED 2   /*polyline dotted linestyle*/
#define TRI_UP 0
#define TRI_DOWN 1

#define NULL_EQ -1 /*signifies no equilibrium price/quantity*/
```

sort: just bubble

```
void sort(int order,int field,int n,Real l[][2])
{ int i,j,swap;
  Real t;

  if((field<0)||(field>1))
  { fprintf(stderr,"\nFail: bad field=%d in sort\n",field);
    exit (0);
  }

  for(i=0;i<n;i++)
  { for(j=0;j<i;j++)
    { if(order)
      { if(l[i][field]>l[j][field]) swap=1; else swap=0; }
      else
      { if(l[i][field]<l[j][field]) swap=1; else swap=0; }

      if(swap)
      { t=l[i][0]; l[i][0]=l[j][0]; l[j][0]=t;
        t=l[i][1]; l[i][1]=l[j][1]; l[j][1]=t;
      }
    }
  }
}
```

xf-polyline: draw a polyline in xfig

```
void xf_polyline(FILE *fp,int lstyle,int lthick,Real dlen,int npoints,
                 int coords[MAX_POINTS][2])
{ int p;

  fprintf(fp,"2 1 %d %d -1 7 0 0 -1 %6.3f 0 0 -1 0 0 %d\n",
          lstyle,lthick,dlen,npoints);
  for(p=0;p<npoints;p++)
  { fprintf(fp,"       %d %d\n",coords[p][0],coords[p][1]); }

}
```

xf-text: draw some text in xfig

```
void xf_text(FILE *fp,int points,Real angle,int x,int y,char text[])
{ fprintf(fp,"4 0 -1 0 0 0 %d %f 4 195 135 %d %d %s\\001\n",
          points,angle,x,y,text);
}
```

xf-triangle: draw a shaded triangle

```
void xf_triangle(FILE *fp,int base_x,int base_y,int peak_y,int dx)
{ int shade;

  if(base_y>peak_y)
  { shade=15;} /*pointing down*/
  else
  { shade=5;}  /*pointing up*/

  fprintf(fp,"2 3 0 1 -1 7 0 0 %d 0.000 0 0 -1 0 0 4\n",shade);
  /*peak*/
  fprintf(fp,"        %d %d  ",base_x+dx/2,peak_y);
  /*base*/
  fprintf(fp,"%d %d  %d %d ",base_x,base_y,base_x+dx,base_y);
  /*back to the peak*/
  fprintf(fp,"%d %d\n",base_x+dx/2,peak_y);
}
```

setcoords: load values into a coordinate pair

```
void setcoords(int c[][2],int p,int x,int y)
{ if(p>=MAX_POINTS)
  { fprintf(stderr,"\nFAIL: p=%d >= MAX_POINTS=%d\n",p,MAX_POINTS); exit(0); }
  c[p][0]=x; c[p][1]=y;
}
```

neat-ticks: find a "neat" inter-tick interval for axis labelling

```
int neat_ticks(int range,int max_ticks)
{ int tick_step,
      tmp=1;
```

```
  if(range>max_ticks)
  { tick_step=(int)(floor(0.5+(range/((Real)(max_ticks)))));
    /*pick a nice stepsize: this is a bit of a kludge*/
    tmp=2;
    if(tick_step>2) tmp=5;
    if(tick_step>7) tmp=10;
    if(tick_step>11) tmp=15;
    if(tick_step>17) tmp=20;
    if(tick_step>22) tmp=25;
    if(tick_step>30) tmp=50;
    if(tick_step>60) tmp=100;
    if(tick_step>120) tmp=150;
    if(tick_step>170) tmp=200;
    if(tick_step>220) tmp=250;
    if(tick_step>300) tmp=500;
    if(tick_step>600) tmp=1000;
  }
  tick_step=tmp;
  return(tick_step);
}
```

draw-axes: do the price and quantity axes dx and dy are returned with the number of pixels in a unit-step miny is baseline y value, maxy is max y value on graph

```
void draw_axes(FILE *fp,int min_q,int max_q,Real min_p,Real max_p,
                int eq_p,int eq_q,int surplus,char fname[],
                int *dx,int *dy,int *miny,int *maxy)
{ int t,p,
      tick,
      start,
      tick_step,
      delta,
      imin_p,imax_p,
      coords[MAX_POINTS][2],
      range;
  char labelstr[MAX_LABELLEN];

  /*draw the axes*/
  p=0;
  setcoords(coords,p++,LMARGIN_X,LMARGIN_Y);
  setcoords(coords,p++,LMARGIN_X,Y_EQ_0);
  setcoords(coords,p++,X_EQ_0,Y_EQ_0);
  xf_polyline(fp,PL_SOLID,AX_THICK,0.00,p,coords);

  /*horizontal axis: quantity*/
  range=(max_q-min_q);
  tick_step=neat_ticks(range,X_TICKS);
  delta=GRAPH_X/(1+(range/tick_step));
  (*dx)=delta/tick_step;
  start=min_q-1;
  if(start<0) start=0;
  for(t=start;t<=max_q;t+=tick_step)
  { tick=LMARGIN_X+(((t-start)/tick_step)*delta);
    p=0;
    setcoords(coords,p++,tick,Y_EQ_0);
    setcoords(coords,p++,tick,Y_EQ_0-TICK_X);
    xf_polyline(fp,PL_SOLID,AX_THICK,0.00,p,coords);
    if(t>min_q-1)
    { sprintf(labelstr,"%d",t);
      xf_text(fp,AX_PTS,0.0,tick,Y_EQ_0+2*TICK_X,labelstr);
    }
  }

  sprintf(labelstr,"Quantity");
  xf_text(fp,AX_PTS,0.0,LMARGIN_X+(int)(GRAPH_X*0.4),Y_EQ_0+4*TICK_X,labelstr);

  /*vertical axis:price*/
  imin_p=(int)(100*min_p);
  imax_p=(int)(100*max_p);
  range=imax_p-imin_p;
  tick_step=neat_ticks(range,Y_TICKS);

  /*fiddle imin-p and i-maxp to make them integer multiples of tick-step*/
  imin_p=(imin_p/tick_step)*tick_step;      /*integer division: lower bound*/
  imax_p=(1+(imax_p/tick_step))*tick_step; /*integer division: upper bound*/
  range=imax_p-imin_p;
  *miny=imin_p;
  *maxy=imax_p;
  delta=GRAPH_Y/(range/tick_step);
  *dy=GRAPH_Y/range;
  delta=(*miny)*(*dy);

  for(t=imin_p;t<=imax_p;t+=tick_step)
  { tick=Y_EQ_0-(t*(*dy))+delta;
    p=0;
    setcoords(coords,p++,LMARGIN_X,tick);
    setcoords(coords,p++,LMARGIN_X+TICK_Y,tick);
    xf_polyline(fp,PL_SOLID,AX_THICK,0.00,p,coords);
    sprintf(labelstr,"%d",t);
    xf_text(fp,AX_PTS,0.0,LMARGIN_X-4*TICK_Y,tick,labelstr);
  }

  /*annotate with key values*/
  if(eq_q==NULL_EQ)
  { sprintf(labelstr,
            "Price                     Eq.Price=<->   Eq.Quant=%2d   Surplus=%4d",
            0,0);
  }
  else
  { sprintf(labelstr,
           "Price                     Eq.Price=%3d   Eq.Quant=%2d   Surplus=%4d",
           eq_p,eq_q,surplus);
  }

  xf_text(fp,AX_PTS,0.0,LMARGIN_X-4*TICK_Y,tick-4*TICK_Y,labelstr);
  /*add the filename for reference, but in small text*/
  sprintf(labelstr,"%s",fname);
  xf_text(fp,AX_PTS/2,0.0,LMARGIN_X-4*TICK_Y,tick-6*TICK_Y,fname);
}
```

supdem: from a list of supplier prices and a list of demander prices, return values: equilibrium price, equilibrium quantity, max surplus. The max surplus figure is integrated from 1 to max-trades, rather than 1 to max(nb,ns) – the maximum total profit that could have been earned is dependent on how many trades are allowed.

```
void supdem(int ns,Agent sellers[],int nb,Agent buyers[],int max_trades,
            Real *ep,int *iq,Real *surplus,int field,char fname[],
            Real *bounds,int verbose)
{ int maxn,a,s,b,no_intersect,not_found,
      q; /*quantity*/
  Real sp[MAX_PRICES][2], /*seller limit and quote prices*/
       bp[MAX_PRICES][2], /*buyer limit and quote prices*/
       profit,tot_surp,
       maxprice,minprice;
  FILE *fp;

  /*these declarations are for the xfig drawing stuff*/
  int p, /*point index*/
      min_q, max_q, /*minimum and maximum quantities on graph*/
      dx,dy,tx,ty,miny,fy,maxy,
      coords[MAX_POINTS][2]; /*coordinate points in polyline etc*/
  char labelstr[MAX_LABELLEN];

  *ep=-1.0;
  *iq=NULL_EQ;

  if(((nb*MAX_UNITS)>MAX_PRICES)||((ns*MAX_UNITS)>MAX_PRICES))
  { fprintf(stderr,"\nFail: too many units in supdem() -- recompile\n");
    exit(0);
  }

  s=0;
  for(a=0;a<ns;a++)
  { if(sellers[a].active)
    { for(q=0;q<sellers[a].quant;q++)
      { sp[s][0]=sellers[a].limit;
        sp[s][1]=sellers[a].price;
        if(s==0)
        { maxprice=sellers[a].price; minprice=sellers[a].limit; }
        else
        { /*for sellers, limit¡=price*/
          if(sellers[a].price>maxprice) maxprice=sellers[a].price;
          if(sellers[a].limit<minprice) minprice=sellers[a].limit;
        }
        s++;
      }
    }
  }

  b=0;
  for(a=0;a<nb;a++)
  { if(buyers[a].active)
    { for(q=0;q<buyers[a].quant;q++)
      { bp[b][0]=buyers[a].limit;
        bp[b][1]=buyers[a].price;
        /*for buyers, limit¿=price*/
        if(buyers[a].limit>maxprice) maxprice=buyers[a].limit;
        if(buyers[a].price<minprice) minprice=buyers[a].price;
        b++;
      }
    }
  }
```

```
sort(1,field,b,bp);
sort(0,field,s,sp);

maxn=(s>b?s:b);
min_q=1;
max_q=maxn;

if(verbose)
{ fprintf(stdout,"Max_trades=%d\n",max_trades);
  fprintf(stdout,"Minprice=%f maxprice=%f min_q=%d max_q=%d\n",
                 minprice,maxprice,min_q,max_q);
}

if(bounds!=NULL)
{ /*autoscaling is OFF*/
  min_q=(int)(*bounds);
  max_q=(int)(*(bounds+1));
  minprice=*(bounds+2);
  maxprice=*(bounds+3);
  if(verbose)
  { fprintf(stdout,"Autoscaling is OFF. Bounds are:\n");
    fprintf(stdout,"Minprice=%f maxprice=%f min_q=%d max_q=%d\n",
                    minprice,maxprice,min_q,max_q);
  }
}

tot_surp=0.0;
if(sp[0][field]>bp[0][field])
{ /*lowest selling price is larger than highest buying price*/
  no_intersect=1;
}

else
{ /*find intersect point*/
  no_intersect=0;
  not_found=1;

  for(q=0;q<maxn;q++)
  { /*intersection?*/
    profit=bp[q][field]-sp[q][field];
    if(not_found)
    {
      if(sp[q][field]>bp[q][field])
      { /*straightforward intersect*/
        *ep=(sp[q-1][field]+bp[q-1][field])/2.0;
        *iq=q;
        not_found=0;
      }

      else
      {
        if((q+1==s)&&(q+1==b))
        { /*last buyer and seller*/
          *ep=(sp[q][field]+bp[q][field])/2.0;
          *iq=q+1;
          if(q<max_trades) tot_surp+=profit;
          not_found=0;
        }
        else
        { if((q+1)==s)
          { /*run out of active sellers but still some buyers*/
            *ep=(bp[q][field]+bp[q+1][field])/2.0;
            *iq=q+1;
            if(q<max_trades) tot_surp+=profit;
            not_found=0;
          }

          else
          { if((q+1)==b)
            { /*run out of active buyers but still some sellers*/
              (*ep)=(sp[q][field]+sp[q+1][field])/2.0;
              (*iq)=q+1;
              if(q<max_trades) tot_surp+=profit;
              not_found=0;
            }
          }
        }
      }

      if(not_found)
      { if(q<max_trades)
        {
          tot_surp+=profit;
        }
      }
    }

    if(verbose)
    { fprintf(stdout,"quantity %2d ",q+1);
      if(q<s) fprintf(stdout,"supply=%5.3f ",sp[q][field]);
      else    fprintf(stdout,"             ");

      if(q<b) fprintf(stdout,"demand=%5.3f ",bp[q][field]);
      else    fprintf(stdout,"             ");

      fprintf(stdout,"profit=%f cum.surp=%f ",profit,tot_surp);
      fprintf(stdout,"\n");
    }
  }
}

*surplus=tot_surp;

if(verbose)
{ switch(field)
  { case EQ_THEORY: fprintf(stdout,"Theoretical");
                    break;
    case EQ_ACTUAL: fprintf(stdout,"Actual");
                    break;
    default: fprintf(stderr,"\nFail: bad field=%d in supdem\n",field);
             exit(0);
  }
  fprintf(stdout," equilibrium price=%f at %d; max surplus=%f\n",
                              *ep,*iq,*surplus);
}

if(fname[0]!='\0')
{ /*write an xfig file*/
  fp=fopen(fname,"w");
```

```
    /*do the preamble*/
    fprintf(fp,"#FIG 3.1\nLandscape\nCenter\nInches\n1200 2\n");

    /*do the axes tickmarks and labelling*/
    draw_axes(fp,min_q,max_q,minprice,maxprice,(int)floor((*ep*100)+0.5),
              *iq,(int)floor((*surplus*100)+0.5),fname,
              &dx,&dy,&miny,&maxy);

    /*do the supply triangles and build the supply curve*/
    p=0;
    for(q=0;q<s;q ++)
    { tx=LMARGIN_X+(q*dx);
      xf_triangle(fp,tx,Y_EQ_0-(int)(sp[q][0]*dy*100)+(miny*dy),
                        Y_EQ_0-(int)(sp[q][1]*dy*100)+(miny*dy),dx);
      fy=Y_EQ_0-(int)(sp[q][field]*dy*100)+(miny*dy);
      setcoords(coords,p++,tx,fy);
      setcoords(coords,p++,tx+dx,fy);
    }
    setcoords(coords,p++,tx+dx,Y_EQ_0-(maxy*dy)+(miny*dy));
    /*do the supply curve*/
    xf_polyline(fp,PL_SOLID,AX_THICK,0.00,p,coords);

    /*do the demand triangles and build the demand curve*/
    p=0;
    for(q=0;q<b;q++)
    { tx=LMARGIN_X+(q*dx);
      xf_triangle(fp,tx,Y_EQ_0-(int)(bp[q][0]*dy*100)+(miny*dy),
                        Y_EQ_0-(int)(bp[q][1]*dy*100)+(miny*dy),dx);
      fy=Y_EQ_0-(int)(bp[q][field]*dy*100)+(miny*dy);
      setcoords(coords,p++,tx,fy);
      setcoords(coords,p++,tx+dx,fy);
    }
    setcoords(coords,p++,tx+dx,Y_EQ_0);
    xf_polyline(fp,PL_SOLID,AX_THICK,0.00,p,coords);

    /*equilibrium price and quantity*/
    if(!no_intersect)
    { p=0;
      setcoords(coords,p++,LMARGIN_X,Y_EQ_0-((*ep)*dy*100)+(miny*dy));
      setcoords(coords,p++,tx+dx,Y_EQ_0-((*ep)*dy*100)+(miny*dy));
      xf_polyline(fp,PL_DASHED,AX_THICK,4.00,p,coords);

      p=0;
      setcoords(coords,p++,LMARGIN_X+((*iq)*dx),Y_EQ_0-((*ep)*dy*100)+(miny*dy));
      setcoords(coords,p++,LMARGIN_X+((*iq)*dx),Y_EQ_0);
      xf_polyline(fp,PL_DASHED,AX_THICK,4.00,p,coords);
    }

    fclose(fp);
  }
}
```

A.8 The Smith Experiments: `smith`

The `main` function allows an experiment defined in a given control-file to be run one or more times. The number of repetitions is a command-line argument, and the control-file parameters and syntax are explained in Section A.3, with an example control-file in Section B.1. Data is gathered and output using the functions defined in `ddat`, `tdat`, and `sd` (Sections A.6, A.5, and A.7 respectively).

The system approximates the parallel asynchronous activity of a community of traders in a discrete cinematographic fashion: time is chunked into discrete 'slices'. At the start of each slice, the active traders that are able to shout are identified and one of them is chosen at random; that agent's price for an offer or bid is then 'shouted', i.e. made public to the remaining agents. The agents in the *other* community (i.e. buyers if a seller shouted, or sellers if a buyer shouted) are then examined to see whether their internal price level is such that they are willing to do a deal. One agent is chosen at random from the list of willing agents to do a deal with the 'shouter', and the trading data of these two agents is adjusted accordingly. If no agents are willing to do a deal, a count of 'failed deals' is incremented and the system loops back to random choice of another shouter. Agents can adjust their price level on the basis of whether the deal succeeded or failed, and whether the shouter came from their community or the other one. This continues until there are either no agents able to trade in one of the communities, or a pre-set number of failed deals is reached.

---

smith.c : the master program
Dave Cliff
Sept 1996

```
#include <math.h>
#include <stdio.h>

#include "max.h"
#include "random.h"
#include "agent.h"
#include "sd.h"
#include "ddat.h"
#include "tdat.h"
#include "expctl.h"
```

reward: monetary reward for a deal

```
Real reward(Agent *a,Real price)
{ Real r;
  if((a->job)==SELL)
  { r=((price-(a->limit))); }
  else
  { r=(((a->limit)-price)); }

  if(r<0.0) r=0.0;

  return(r);
}
```

get-price: get a price from an agent)

```
Real get_price(Agent *a,int id,int random,int verbose)
{ Real price;
  Real rmin=0.01,rmax=4.0;  /*bounds on random prices*/
  if(random)
  { /*agent price is generated at random*/
    if(rmax<a->limit)
    { fprintf(stderr,"\nFail: rmax too low in get_price()\n");
      exit(0);
    }

    if(a->job==BUY) price=rmin+randval((a->limit)-rmin);
    else price=(a->limit)+randval(rmax-(a->limit));
    price=(floor(0.5+(price*100)))/100;
    a->price=price;
  }
  else price=a->price;

  if(verbose)
  { if(a->job==BUY) fprintf(stdout,"Buyer %d bids at %5.3f (reward=%5.3f)\n",
                            id,price,reward(a,price));
    else fprintf(stdout,"Seller %d offers at %5.3f (reward=%5.3f)\n",
                 id,price,reward(a,price));
  }
  return(price);
}
```

get-willing: form a list of agents willing to deal

```
int get_willing(Real price,Agent agents[],int n,int ilist[],char *s,int random,
                int verbose)
{ int willing=0,a;
  Real r_price,p;

  p=price;

  for(a=0;a<n;a++)
  { if(random)
    { /*agent generates a price at random, compares it to given price*/
      /*and is willing if random price makes a profit*/
      agents[a].willing=0;

      if(agents[a].active)
      { r_price=get_price(agents+a,a,random,verbose);
        if(agents[a].job==BUY)
        { if(r_price>price)
          { agents[a].willing=1; p=r_price; }
        }

        else
        { if(r_price<price)
          { agents[a].willing=1; p=r_price; }
        }
      }
    }

    else
    { /*use some intelligence*/
      willing_trade(agents+a,price);
    }

    if(agents[a].willing)
    { ilist[willing]=a;
      willing++;
      if(verbose)
      { fprintf(stdout,"%s%2d willing (r)price=%5.3f reward=%5.3f\n",
                 s,a,p,reward(agents+a,price));
      }
    }
  }
  if(verbose) fprintf(stdout,"%d traders willing to deal\n",willing);

  return(willing);
}
```

get-able: form a list of agents able to deal

```
int get_able(Real price,Agent agents[],int n,int ilist[],char *s,int verbose)
{ int able=0,a;

  for(a=0;a<n;a++)
  { if(agents[a].able)
    { ilist[able]=a;
      able++;
      if(verbose)
      { fprintf(stdout,"%s%2d able (reward=%5.3f)\n",
                 s,a,reward(agents+a,price));
      }
    }
  }
  return(able);
}
```

bank: adjust bank balances of buyer and seller in a deal

```
void bank(Agent *s,Agent *b,Real price,Real *surplus,int verbose)
{ Real r;

  /*seller*/
  r=reward(s,price);
  (s->bank)+=r;
  (s->a_gain)+=r;
  (*surplus)+=(r);

  (s->quant)--;
  if(s->quant<1) s->active=0;
  if(verbose)
  { fprintf(stdout,"Seller: limit=%f reward=%f bank=%f quant=%d (surp=%f)\n",
                   s->limit,r,s->bank,s->quant,*surplus);
  }

  /*buyer*/
  r=reward(b,price);
  (b->bank)+=r;
  (b->a_gain)+=r;
  (*surplus)+=(r);

  (b->quant)--;
  if(b->quant<1) b->active=0;
  if(verbose)
  { fprintf(stdout,"Buyer: limit=%f reward=%f bank=%f quant=%d (surp=%f)\n",
                    b->limit,r,b->bank,b->quant,*surplus);
  }
}
```

day-init: initialise all data structures for start of day

```
void day_init(int exp_number,int day_number,Day_data *ddat,Expctl *ec,
              Agent sellers[],Agent buyers[],
              Real *p_0,Real *max_surplus,int verbose)
{ int b,s,q_0,s_sched,d_sched,n_buy,n_sell;
  Real eq_profit;
  char filename[40];

  /*initialise the buyers*/

  if(day_number==0)
  { /*first day: read the first demand schedule*/
    ec->d_sched=0;
  }
  else
  if((day_number-1)==(ec->dem_sched[ec->d_sched].last_day))
  { /*previous day was last day on that demand schedule: update*/
    (ec->d_sched)++;
    if(ec->d_sched==ec->n_dem_sched)
    { fprintf(stderr,"\nFail: ran out of demand schedules on day %d\n",
              day_number);
      exit(0);
    }
  }
  d_sched=ec->d_sched;
  n_buy=ec->dem_sched[d_sched].n_agents;

  /*mark all buyers active, set quantities and limit prices*/
  for(b=0;b<n_buy;b++)
  { buyers[b].quant=ec->dem_sched[d_sched].agents[b].n_units;
    buyers[b].active=1;
    buyers[b].a_gain=0.0;
    /*NOTE: ONLY ALLOWS FOR ONE LIMIT PRICE*/
    buyers[b].limit=ec->dem_sched[d_sched].agents[b].limit[0];
    set_price(buyers+b);
    if(verbose) fprintf(stdout,"buyer %d price %f\n",b,buyers[b].price);
  }

  /*initialise the sellers*/

  if(day_number==0)
  { /*first day: read the first demand schedule*/
    ec->s_sched=0;
  }
  else
  if((day_number-1)==(ec->sup_sched[ec->s_sched].last_day))
  { /*previous day was last day on that supply schedule: update*/
    (ec->s_sched)++;
    if(ec->s_sched==ec->n_sup_sched)
    { fprintf(stderr,"\nFail: ran out of supply schedules on day %d\n",
              day_number);
      exit(0);
    }
  }
  s_sched=ec->s_sched;
  n_sell=ec->sup_sched[s_sched].n_agents;

  /*mark all sellers active, set quantities and limit prices*/
  for(s=0;s<n_sell;s++)
  { sellers[s].quant=ec->sup_sched[s_sched].agents[s].n_units;
```

```
    sellers[s].active=1;
    sellers[s].a_gain=0.0;
    /*NOTE: ONLY ALLOWS FOR ONE LIMIT PRICE*/
    sellers[s].limit=ec->sup_sched[s_sched].agents[s].limit[0];
    set_price(sellers+s);
    if(verbose) fprintf(stdout,"seller %d price %f\n",s,sellers[s].price);
  }

  /*find theoretical equilibrium price*/
  if(exp_number==0) sprintf(filename,"%ssd%02d_000.fig",ec->id,day_number+1);
  else sprintf(filename,"\0");
  supdem(n_sell,sellers,n_buy,buyers,
         ec->max_trades,p_0,&q_0,max_surplus,EQ_THEORY,filename,
         NULL,verbose);

  /*set theoretical gains for buyers and sellers*/
  for(b=0;b<n_buy;b++)
  { eq_profit=buyers[b].quant*(buyers[b].limit-(*p_0));
    if(eq_profit<0.0) eq_profit=0.0;
    buyers[b].t_gain=eq_profit;
  }
  for(s=0;s<n_sell;s++)
  { eq_profit=sellers[s].quant*((*p_0)-sellers[s].limit);
    if(eq_profit<0.0) eq_profit=0.0;
    sellers[s].t_gain=eq_profit;
  }
}
```

trade: see if a buyer and a seller can be found who will enter into a trade

```
void trade(Trade_data *tdat,Agent sellers[],Agent buyers[],Expctl *ec,
           Real max_surplus,Real *surplus,int *stat,int verbose)
{ int b,s,         /*buyer and seller indices*/
      dt,          /*deal type*/
      status,      /*what's happening*/
      eq_q,        /*equilibrium quantity*/
      n_willing,   /*number of agents willing to trade at a given price*/
      n_able,      /*number of agents able to trade at a given price*/
      n_fails,     /*number of failed/declined bids/offers*/
      n_buy,       /*number of buyers*/
      n_sell,      /*number of sellers*/
      active_b,    /*number of active buyers*/
      active_s,    /*number of active sellers*/
      sell_shout,  /*can sellers shout offers?*/
      buy_shout,   /*can buyers shout bids?*/
      traders,     /*number of traders to choose from when generating shout*/
      first_offer,/*flag raised until an opening offer is made*/
      first_bid,   /*falg raised unitl an opening bid is made*/
      ilist[MAX_AGENTS]; /*list of indices*/

  Real eq_p,       /*equilibrium price*/
       cur_surp,   /*current actual max surplus*/
       best_offer,/*used in NYSE rules*/
       best_bid,   /*used in NYSE rules*/
       price;      /*price of bid/ask*/

  n_buy=ec->dem_sched[ec->d_sched].n_agents;
  n_sell=ec->sup_sched[ec->s_sched].n_agents;
  sell_shout=ec->sup_sched[ec->s_sched].can_shout;
  buy_shout=ec->dem_sched[ec->d_sched].can_shout;
```

```
if((sell_shout==0)&&(buy_shout==0))
{ fprintf(stderr,"\nFAIL: Can't have both buyers AND sellers silent\n");
  exit(0);
}

/*find the theoretical equilibrium price*/
supdem(n_sell,sellers,n_buy,buyers,ec->max_trades,
       &eq_p,&eq_q,&cur_surp,EQ_THEORY,
       "\0",NULL,verbose);
if(eq_q!=NULL_EQ) { tdat->t_eq_p=eq_p; tdat->t_eq_q=eq_q; }
else tdat->t_eq_q=NULL_EQ;

/*find the actual equilibrium price*/
supdem(n_sell,sellers,n_buy,buyers,ec->max_trades,
       &eq_p,&eq_q,&cur_surp,EQ_ACTUAL,
       "\0",NULL,verbose);
if(eq_q!=NULL_EQ) { tdat->a_eq_p=eq_p; tdat->a_eq_q=eq_q; }
else tdat->a_eq_q=NULL_EQ;

n_fails=0;
status=NO_DEAL;
first_offer=1; first_bid=1;
while((status==NO_DEAL)&&(n_fails<MAX_FAILS))
{
  /*count active agents and mark them as able to bid*/
  active_b=0;
  for(b=0;b<n_buy;b++)
  { if(buyers[b].active)
    { buyers[b].able=1;
      active_b++;
    }
    else buyers[b].able=0;
  }
  active_s=0;
  for(s=0;s<n_sell;s++)
  { if(sellers[s].active)
    { sellers[s].able=1;
      active_s++;
    }
    else sellers[s].able=0;
  }

  traders=0;
  if(sell_shout) traders+=active_s;
  if(buy_shout) traders+=active_b;

  if(verbose) fprintf(stdout,"%d traders: active_s=%d active_b=%d\n",
                      traders,active_s,active_b);

  if(irand(traders)<active_s)
  { /*is there a seller able to make an offer?*/
    dt=OFFER;

    if(ec->nyse&&(!first_offer))
    { if(ec->random)
      { /*any seller with a limit price higher than best offer can't deal*/
        for(s=0;s<n_sell;s++)
        { if(sellers[s].limit>best_offer) sellers[s].able=0; }
      }
    else
    { /*any seller with an equal or higher price can't offer*/
      for(s=0;s<n_sell;s++)
      { if(sellers[s].price>=best_offer) sellers[s].able=0; }
    }
  }
  n_able=get_able(0.0,sellers,n_sell,ilist,"S",verbose);

  if(n_able>0)
  { /*an able seller makes an offer*/
    s=ilist[irand(n_able)];
    /*get price for seller*/
    price=get_price(sellers+s,s,ec->random,verbose);
    if(ec->nyse)
    { if(first_offer)
      { best_offer=price; first_offer=0; }
      else
      { if(price<best_offer) best_offer=price; }
    }

    /*get willing buyers*/
    n_willing=get_willing(price,buyers,n_buy,ilist,"B",ec->random,verbose);
    if(n_willing>0) status=DEAL;
  }
  else
  {
    if(verbose) fprintf(stdout,"No sellers able to offer\n");
    n_fails=MAX_FAILS;
    status=END_DAY;
  }
}

else
{ /*is there a buyer able to make a bid?*/
  dt=BID;
  if(ec->nyse&&(!first_bid))
  { if(ec->random)
    { /*any buyer with limit lower than best bid can't deal*/
      for(b=0;b<n_buy;b++)
      { if(buyers[b].limit<best_bid) buyers[b].able=0; }
    }
    else
    { /*any buyer with an equal or lower price can't bid*/
      for(b=0;b<n_buy;b++)
      { if(buyers[b].price<=best_bid) buyers[b].able=0; }
    }
  }
  n_able=get_able(0.0,buyers,n_buy,ilist,"B",verbose);

  if(n_able>0)
  { /*an able buyer makes a bid*/
    b=ilist[irand(n_able)];

    /*get price for buyer*/
    price=get_price(buyers+b,b,ec->random,verbose);
    if(ec->nyse)
    { if(first_bid)
      { best_bid=price; first_bid=0; }
      else
      { if(price>best_bid) best_bid=price; }
    }
```

```
        /*get willing selllers*/
        n_willing=get_willing(price,sellers,n_sell,ilist,"S",ec->random,verbose);
        if(n_willing>0) status=DEAL;
      }
      else
      { if(verbose) fprintf(stdout,"No buyers able to bid\n");
        n_fails=MAX_FAILS;
        status=END_DAY;
      }
    }

    if(status==DEAL)
    { /*DEAL*/
      if(dt==OFFER)
      { /*select the willing buyer for this offer*/
        b=ilist[irand(n_willing)];
        if(verbose)
        { fprintf(stdout,
                  "Seller %d sells to Buyer %d (reward=%5.3f)\n",
                  s,b,reward(buyers+b,price));
        }
      }
      else
      { /*select the willing seller for this bid*/
        s=ilist[irand(n_willing)];
        if(verbose)
        { fprintf(stdout,
                  "Buyer %d buys from Seller %d (reward=%5.3f)\n",
                  b,s,reward(sellers+s,price));
        }
      }

      /*record what happened*/
      tdat->deal_p=price;
      tdat->deal_t=dt;

      /*update trading strategies of buyers and sellers*/
      shout_update(dt,status,n_sell,sellers,n_buy,buyers,price,verbose);

      /*update bank accounts of buyer and seller*/
      bank(sellers+s,buyers+b,price,surplus,verbose);
    }
    else
    { /*NO DEAL or END DAY*/
      n_fails++;
      if(verbose) fprintf(stdout,"No willing takers (fails=%d)\n",n_fails);
      tdat->deal_p=-1.0; /*negative price =¿ no deal*/

      /*update trading strategies of buyers and sellers*/
      shout_update(dt,status,n_sell,sellers,n_buy,buyers,price,verbose);
    }
  }
  *stat=status;
}
```

```
int main(int argc,char *argv[])
{ int n_trans,    /*number of transactions on a day*/
      d,          /*day*/
      status,     /*how things are going*/
      rs,         /*random seed*/
      s,b,        /*seller, buyer, loop indices*/
      n_buy,      /*number of buyers*/
      n_sell,     /*number of sellers*/
      n_trades,   /*number of trades done in a day*/
      n_exps,     /*number of experiments to run*/
      max_trades, /*maxmimum number of trades in a session*/
      verbose=0,
      ats_n[MAX_TRADES], /*counts of entries in ats[]*/
      eq,         /*equilibrium quantity*/
      dummy_i,    /*dummy integer*/
      e,          /*experiment number*/
      t;          /*transaction number within a day*/
  Real price,p_0,sigmasum,alpha,ep,last_price,sum_price_diff,
       dummy_r1,dummy_r2,
       sum_price,
       diff,pd,pdisp,pds,
       alphatrans,       /*alpha over transaction sequence (cf G+S fig6)*/
       ats[MAX_TRADES], /*alpha over transaction sequence (cf G+S fig6)*/
       bounddata[4],    /*can be used to inhibit autoscaling on supdem*/
       *bounds,
       max_surplus,surplus,efficiency;
  char fname[60];
  Day_data ddat[MAX_N_DAYS];
  Trade_data tdat[MAX_N_DAYS][MAX_TRADES];
  Real_stat ats_e[MAX_TRADES]; /*for summarising ats[] over experiments*/
  Agent buyers[MAX_AGENTS],
            sellers[MAX_AGENTS];
  Expctl expctl;
  FILE *fp;

  if(argc<3)
  { fprintf(stderr,"\nUsage: smith <n_exps> <datafilename>\n");
    exit(0);
  }
  sscanf(argv[1],"%d",&n_exps);
  fprintf(stdout,"%d experiments, data-file=%s\n",n_exps,argv[2]);

  if(n_exps==1) rs=0;
  else rs=999;

  rseed(&rs);

  expctl_in(argv[2],&expctl,1);

  /*initialise daily data records*/
  for(d=0;d<expctl.n_days;d++) ddat_init(ddat+d);

  for(t=0;t<MAX_TRADES;t++)
  { ats_n[t]=0; ats[t]=0.0;
    ats_e[t].n=0; ats_e[t].sum=0.0; ats_e[t].sumsq=0.0;
  }

  for(e=0;e<n_exps;e++)
  { /*do one experiment*/
```

```
buy_init(buyers,verbose);
sell_init(sellers,verbose);

for(d=0;d<expctl.n_days;d++)
{ /*one trading period or "day"*/

  /*set maximum number of trades in this day*/
  max_trades=expctl.max_trades;
  if(verbose) fprintf(stdout,"\nday %d: %d trades\n",d+1,max_trades);

  /*set things up for the start of the day*/
  day_init(e,d,ddat+d,&expctl,sellers,buyers,&p_0,&max_surplus,verbose);
  n_buy=expctl.dem_sched[expctl.d_sched].n_agents;
  n_sell=expctl.sup_sched[expctl.s_sched].n_agents;

  surplus=0.0;
  n_trades=0;
  sigmasum=0.0;
  sum_price=0.0;
  sum_price_diff=0.0;

  bounds=NULL;

bounddata[0]=1; bounddata[1]=12; bounddata[2]=0.0; bounddata[3]=3.75; bounds=&(bounddata[0]);

  if(e==0) /*first experiment?*/
  { /*write a figure of the actual supply and demand curves*/
    sprintf(fname,"%ssd%02d_%03d_000.fig",expctl.id,d+1,n_trades+1);
    supdem(n_sell,sellers,n_buy,buyers,max_trades,
           &dummy_r1,&dummy_i,&dummy_r2,
           EQ_ACTUAL,fname,bounds,verbose);
  }

  for(t=0;t<max_trades;t++)
  { /*one trading session: either a trade occurs or a fail is recorded*/

    if(verbose) fprintf(stdout,"\nday %d trade %d\n",d,t+1);

    trade(&(tdat[d][t]),sellers,buyers,&expctl,
          max_surplus,&surplus,&status,verbose);

    /*this can generate *lots* of data-files*/
    if((verbose>0)&&(e==0)) /*first experiment?*/
    { /*print a figure of the actual supply and demand curves*/
      sprintf(fname,
              "%ssd%02d_%03d_%03d.fig",expctl.id,d+1,n_trades+1,t+1);
      fprintf(stdout,"Writing %s\n",fname);
      supdem(n_sell,sellers,n_buy,buyers,max_trades,
             &dummy_r1,&dummy_i,&dummy_r2,
             EQ_ACTUAL,fname,bounds,verbose);
    }

    /*calculate stats*/
    if(status==DEAL)
    { if(t>0) last_price=price;
      price=tdat[d][t].deal_p;
      if(t>0) sum_price_diff+=((price-last_price)*(price-last_price));
      pds=((price-p_0)*(price-p_0));
      (ats[n_trades])+=pds;
      (ats_n[n_trades])++;
      n_trades++;
      sum_price+=price;
      sigmasum+=pds;
      alpha=(100*sqrt(sigmasum/n_trades))/p_0;
      efficiency=(surplus/max_surplus)*100;
      if(verbose)
      { fprintf(stdout,"Day %d deal %d alpha=%f efficiency=%f\n",
                         d,n_trades,alpha,efficiency);
      }
    }
    else
    { if(status==END_DAY) /*give up*/
      t=max_trades;
    }
  } /*end of the trading session*/

  /*update the data for this day*/
  /*profit dispersion*/
  pd=0.0;
  for(b=0;b<n_buy;b++)
  { diff=((buyers[b].a_gain)-(buyers[b].t_gain));
    pd+=(diff*diff);
  }
  for(s=0;s<n_sell;s++)
  { diff=((sellers[s].a_gain)-(sellers[s].t_gain));
    pd+=(diff*diff);
  }
  pdisp=sqrt((1/((Real)(n_buy+n_sell)))*pd);
  if(verbose) fprintf(stdout,"Dispersion=%f\n",pdisp);

  ddat_update(ddat+d,n_trades,sum_price,alpha,pdisp,efficiency,
              sum_price_diff);

} /*end of the day loop*/

if(e==0)
{ /*plot the trade stats in xgraph format*/
  sprintf(fname,"%sresults.xg",expctl.id);
  xg_trades_graph(tdat,ddat,expctl.n_days,max_trades,fname,n_exps);

  /*plot this exp's per-trans rms deviation of deal price from equilib*/
  sprintf(fname,"%sres_rms.xg",expctl.id);
  fp=fopen(fname,"w");
  for(t=0;t<max_trades;t++)
  { if(ats_n[t]>0)
    { fprintf(fp,"%d ",t+1);
      fprintf(fp,"%f \n",sqrt(ats[t]/ats_n[t]));
    }
  }
  fclose(fp);
}

for(t=0;t<max_trades;t++)
{ if(ats_n[t]>0)
  { alphatrans=sqrt(ats[t]/ats_n[t]);
    (ats_e[t].sum)+=alphatrans;
    (ats_e[t].sumsq)+=(alphatrans*alphatrans);
```

```
        (ats_e[t].n)++;
      }
    }

    fprintf(stdout,"experiment %d done\n",e);

  } /*end of the experiment loop*/

  /*plot the end-of-day stats in xgraph format*/
  sprintf(fname,"%sres_day.xg",expctl.id);
  xg_daily_graph(ddat,expctl.n_days,n_exps,fname);

  /*plot per-trans rms deviation of deal price from equilib, over exps*/
  sprintf(fname,"%sres_rms_avg.xg",expctl.id);
  fp=fopen(fname,"w");
  fprintf(fp,"TitleText: %s: n=%d\n\n",fname,n_exps);
  /*mean*/
  fprintf(fp,"\"Mean\n");
  for(t=0;t<max_trades;t++)
  { if(ats_e[t].n>0)
    { fprintf(fp,"%d ",t+1);
      fprintf(fp,"%f \n",ats_e[t].sum/ats_e[t].n);
    }
  }
  fprintf(fp,"\n");
  /*+1 standard dev*/
  fprintf(fp,"\"Mean+1sd\n");
  for(t=0;t<max_trades;t++)
  { if(ats_e[t].n>0)
    { fprintf(fp,"%d ",t+1);
      alphatrans=ats_e[t].sum/ats_e[t].n;
      fprintf(fp,"%f \n",
        alphatrans+sqrt((ats_e[t].sumsq/ats_e[t].n)-(alphatrans*alphatrans)));
    }
  }
  fprintf(fp,"\n");
  /*-1 standard dev*/
  fprintf(fp,"\"Mean-1sd\n");
  for(t=0;t<max_trades;t++)
  { if(ats_e[t].n>0)
    { fprintf(fp,"%d ",t+1);
      alphatrans=ats_e[t].sum/ats_e[t].n;
      fprintf(fp,"%f \n",
        alphatrans-sqrt((ats_e[t].sumsq/ats_e[t].n)-(alphatrans*alphatrans)));
    }
  }
  fprintf(fp,"\n");
  /*n as a proportion of nexps*/
  fprintf(fp,"\"n/n_exps\n");
  for(t=0;t<max_trades;t++)
  { if(ats_e[t].n>0)
    { fprintf(fp,"%d ",t+1);
      fprintf(fp,"%f \n",((Real)ats_e[t].n)/n_exps);
    }
  }
  fclose(fp);
  return(1);
}
```

A.9 Putting it together: `makefile`

This makefile can be used with the Unix system utility **`make`** to create an executable called **`smith`**. Also, typing **`make clean`** will erase the executable and all object files.

```
OBJS = random.o sd.o agent.o tdat.o ddat.o expctl.o
LIBS = -lm
HDRS = sd.h agent.h tdat.h ddat.h max.h expctl.h \
       random.h
# CFLAGS = -ggdb
CFLAGS =  -O -Aa
CC = cc

smith : smith.o ${OBJS} ; ${CC} ${CFLAGS} smith.o ${OBJS} ${LIBS} -o $@

expctl.o: max.h random.h expctl.h

sd.o: random.h agent.h max.h

agent.o: random.h agent.h

tdat.o: random.h agent.h max.h tdat.h

ddat.o: random.h agent.h max.h ddat.h

smith.o : random.h agent.h max.h ddat.h tdat.h expctl.h

random.o : random.h

.o: ${HDRS} ; ${CC} -c ${CFLAGS} $<

clean: ; rm *.o smith
```

B Sample Input and Output

An experiment control file, `zip1hii.dat` is shown in Section B.1. This defines a simple experiment where each agent buys or sells only one unit, and the supply and demand schedules are fixed for the first ten days, after which both supply and demand are increased and the experiment continues for another five days. This is the control file used to run the increased-demand experiments, results from which were illustrated in Figures 46 and 47.

The command-line argument `smith 5 zip1hi.dat` was given to the Unix prompt, and the resultant output from the system (with `verbose=0`) is shown in Section B.2.

The following files were created during this run:

```
-rw-------   1 davec    davec       3566 Jun  4 14:49 zip1hires_day.xg
-rw-------   1 davec    davec         84 Jun  4 14:49 zip1hires_rms.xg
-rw-------   1 davec    davec        413 Jun  4 14:49 zip1hires_rms_avg.xg
-rw-------   1 davec    davec       4553 Jun  4 14:49 zip1hiresults.xg
-rw-------   1 davec    davec       5901 Jun  4 14:49 zip1hisd01_000.fig
-rw-------   1 davec    davec       5905 Jun  4 14:49 zip1hisd01_001_000.fig
-rw-------   1 davec    davec       6409 Jun  4 14:49 zip1hisd02_000.fig
-rw-------   1 davec    davec       6413 Jun  4 14:49 zip1hisd02_001_000.fig
-rw-------   1 davec    davec       6409 Jun  4 14:49 zip1hisd03_000.fig
-rw-------   1 davec    davec       6413 Jun  4 14:49 zip1hisd03_001_000.fig
-rw-------   1 davec    davec       6409 Jun  4 14:49 zip1hisd04_000.fig
-rw-------   1 davec    davec       6413 Jun  4 14:49 zip1hisd04_001_000.fig
-rw-------   1 davec    davec       6409 Jun  4 14:49 zip1hisd05_000.fig
-rw-------   1 davec    davec       6413 Jun  4 14:49 zip1hisd05_001_000.fig
-rw-------   1 davec    davec       6409 Jun  4 14:49 zip1hisd06_000.fig
-rw-------   1 davec    davec       6413 Jun  4 14:49 zip1hisd06_001_000.fig
-rw-------   1 davec    davec       6409 Jun  4 14:49 zip1hisd07_000.fig
-rw-------   1 davec    davec       6413 Jun  4 14:49 zip1hisd07_001_000.fig
-rw-------   1 davec    davec       6408 Jun  4 14:49 zip1hisd08_000.fig
-rw-------   1 davec    davec       6412 Jun  4 14:49 zip1hisd08_001_000.fig
-rw-------   1 davec    davec       6408 Jun  4 14:49 zip1hisd09_000.fig
-rw-------   1 davec    davec       6260 Jun  4 14:49 zip1hisd09_001_000.fig
-rw-------   1 davec    davec       6408 Jun  4 14:49 zip1hisd10_000.fig
-rw-------   1 davec    davec       6412 Jun  4 14:49 zip1hisd10_001_000.fig
-rw-------   1 davec    davec       6664 Jun  4 14:49 zip1hisd11_000.fig
-rw-------   1 davec    davec       6668 Jun  4 14:49 zip1hisd11_001_000.fig
-rw-------   1 davec    davec       6665 Jun  4 14:49 zip1hisd12_000.fig
-rw-------   1 davec    davec       6669 Jun  4 14:49 zip1hisd12_001_000.fig
-rw-------   1 davec    davec       6665 Jun  4 14:49 zip1hisd13_000.fig
-rw-------   1 davec    davec       6517 Jun  4 14:49 zip1hisd13_001_000.fig
-rw-------   1 davec    davec       6665 Jun  4 14:49 zip1hisd14_000.fig
-rw-------   1 davec    davec       6669 Jun  4 14:49 zip1hisd14_001_000.fig
-rw-------   1 davec    davec       6665 Jun  4 14:49 zip1hisd15_000.fig
-rw-------   1 davec    davec       6669 Jun  4 14:49 zip1hisd15_001_000.fig
```

The files matching `*.fig` are illustrations of the supply and demand at the start of each day's trading: the contents of `zip1hisd01_001_000.fig` are shown in Section B.3: when files such as this are used as input to `Xfig`, graphics such as those shown in Figures 57, 58, and 59 are produced.

The files matching `*.xg` produce data for X–Y graphs: `zip1hiresults.xg` is the time series of transaction prices from the first of the $n = 5$ experiments (see e.g. Figure 46); `zip1hires_rms.xg` is the root mean square (RMS) transaction price deviation from equilibrium price indexed by transaction number for the first experiment, while `zip1hires_rms_avg.xg` is the average RMS data for the $n = 5$ experiments (see e.g. Figures 32 to 35); finally, `zip1hires_day.xg` contains time series (indexed by day number) of various statistics calculated at the end of each trading day, such as average transaction price, Smith's α measure, allocative efficiency, profit dispersion, etc.. The contents of `zip1hires_day.xg` are shown in Section B.4

B.1 Control File

```
#id string
zip1hi
#number of days
15
#min trades per day
9
#max trades per day
9
#random flag: 0 => intelligent traders; 1 => ZI-C
0
#nyse flag (1=> on)
0
#number of demand schedules
2
#first demand schedule starts here
#number of agents
11
#start day
0
#end day
9
#can_shout
1
#for each agent: number of units, price of each unit
1 3.25
1 3.00
1 2.75
1 2.50
1 2.25
1 2.00
1 1.75
1 1.50
1 1.25
1 1.00
1 0.75
#end schedule
#second demand schedule starts here -- all prices up by 0.50
#number of agents
11
#start day
10
#end day
14
#can_shout
1
#for each agent: number of units, price of each unit
1 3.75
1 3.50
1 3.25
1 3.00
```

```
1 2.75
1 2.50
1 2.25
1 2.00
1 1.75
1 1.50
1 1.25
#end demand schedule
#number of supply schedules
1
#first supply schedule starts here
#number of agents
11
#start day
0
#end day
14
#can_shout
1
#for each agent: number of units, price of each unit
1 0.75
1 1.00
1 1.25
1 1.50
1 1.75
1 2.00
1 2.25
1 2.50
1 2.75
1 3.00
1 3.25
#end schedule
#second supply schedule starts here
#number of agents
11
#start day
10
#end day
14
#can_shout
1
#for each agent: number of units, price of each unit -- all up by 0.50
1 1.25
1 1.50
1 1.75
1 2.00
1 2.25
1 2.50
1 2.75
1 3.00
1 3.25
1 3.50
1 3.75
#end schedule
```

B.2 Output to stdout

```
5 experiments, data-file=zip1hi.dat

: Seed is 999
ID: zip1hi
15 days: min_trades=9 max_trades=9
Intelligent traders; no NYSE rules
2 demand schedules:
  Demand schedule 0:
  11 agents: from day 0 to day 9
(These traders CAN SHOUT)
  Agent  0, 1 units: 3.250000
  Agent  1, 1 units: 3.000000
  Agent  2, 1 units: 2.750000
  Agent  3, 1 units: 2.500000
  Agent  4, 1 units: 2.250000
  Agent  5, 1 units: 2.000000
  Agent  6, 1 units: 1.750000
  Agent  7, 1 units: 1.500000
  Agent  8, 1 units: 1.250000
  Agent  9, 1 units: 1.000000
  Agent 10, 1 units: 0.750000
  Demand schedule 1:
  11 agents: from day 10 to day 14
(These traders CAN SHOUT)
  Agent  0, 1 units: 3.750000
  Agent  1, 1 units: 3.500000
  Agent  2, 1 units: 3.250000
  Agent  3, 1 units: 3.000000
  Agent  4, 1 units: 2.750000
  Agent  5, 1 units: 2.500000
  Agent  6, 1 units: 2.250000
  Agent  7, 1 units: 2.000000
  Agent  8, 1 units: 1.750000
  Agent  9, 1 units: 1.500000
  Agent 10, 1 units: 1.250000
1 supply schedules:
  Supply schedule 0:
  11 agents: from day 0 to day 14
(These traders CAN SHOUT)
  Agent  0, 1 units: 0.750000
  Agent  1, 1 units: 1.000000
  Agent  2, 1 units: 1.250000
  Agent  3, 1 units: 1.500000
  Agent  4, 1 units: 1.750000
  Agent  5, 1 units: 2.000000
  Agent  6, 1 units: 2.250000
  Agent  7, 1 units: 2.500000
  Agent  8, 1 units: 2.750000
  Agent  9, 1 units: 3.000000
  Agent 10, 1 units: 3.250000
experiment 0 done
experiment 1 done
experiment 2 done
experiment 3 done
experiment 4 done
```

B.3 Sample Supply-Demand `.fig` Output File and Graphic

```
#FIG 3.1
Landscape
Center
Inches
1200 2
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 3
	 1260 1080
	 1260 6119
	 7139 6119
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 6119
	 1260 5994
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1794 6119
	 1794 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 1794 6369 1\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 2328 6119
	 2328 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 2328 6369 2\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 2862 6119
	 2862 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 2862 6369 3\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 3396 6119
	 3396 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 3396 6369 4\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 3930 6119
	 3930 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 3930 6369 5\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 4464 6119
	 4464 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 4464 6369 6\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 4998 6119
	 4998 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 4998 6369 7\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 5532 6119
	 5532 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 5532 6369 8\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 6066 6119
	 6066 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 6066 6369 9\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 6600 6119
	 6600 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 6600 6369 10\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 7134 6119
	 7134 5994
4 0 -1 0 0 0 18 0.000000 4 195 135 7134 6369 11\001
4 0 -1 0 0 0 18 0.000000 4 195 135 3611 6619 Quantity\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 6119
	 1406 6119
4 0 -1 0 0 0 18 0.000000 4 195 135 676 6119 50\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 5419
	 1406 5419
4 0 -1 0 0 0 18 0.000000 4 195 135 676 5419 100\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 4719
	 1406 4719
4 0 -1 0 0 0 18 0.000000 4 195 135 676 4719 150\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 4019
	 1406 4019
4 0 -1 0 0 0 18 0.000000 4 195 135 676 4019 200\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 3319
	 1406 3319
4 0 -1 0 0 0 18 0.000000 4 195 135 676 3319 250\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 2619
	 1406 2619
4 0 -1 0 0 0 18 0.000000 4 195 135 676 2619 300\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 1919
	 1406 1919
4 0 -1 0 0 0 18 0.000000 4 195 135 676 1919 350\001
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 2
	 1260 1219
	 1406 1219
4 0 -1 0 0 0 18 0.000000 4 195 135 676 1219 400\001
4 0 -1 0 0 0 18 0.000000 4 195 135 676 635 Price                     Eq.Price=176   Eq.Quant= 4   Surplus= 398\001
```

```
4 0 -1 0 0 0 9 0.000000 4 195 135 676 343 zip1hisd01_001_000.fig\001
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 1527 5503  1260 5769  1794 5769 1527 5503
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 2061 5307  1794 5419  2328 5419 2061 5307
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 2595 4747  2328 5069  2862 5069 2595 4747
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 3129 4383  2862 4719  3396 4719 3129 4383
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 3663 3907  3396 4369  3930 4369 3663 3907
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 4197 3557  3930 4019  4464 4019 4197 3557
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 4731 2983  4464 3669  4998 3669 4731 2983
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 5265 2535  4998 3319  5532 3319 5265 2535
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 5799 2479  5532 2969  6066 2969 5799 2479
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 6333 1863  6066 2619  6600 2619 6333 1863
2 3 0 1 -1 7 0 0 15 0.000 0 0 -1 0 0 4
	 6867 1373  6600 2269  7134 2269 6867 1373
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 23
	1260 5503
	1794 5503
	1794 5307
	2328 5307
	2328 4747
	2862 4747
	2862 4383
	3396 4383
	3396 3907
	3930 3907
	3930 3557
	4464 3557
	4464 2983
	4998 2983
	4998 2535
	5532 2535
	5532 2479
	6066 2479
	6066 1863
	6600 1863
	6600 1373
	7134 1373
	7134 1219
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 1527 2928  1260 2269  1794 2269 1527 2928
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 2061 3250  1794 2969  2328 2969 2061 3250
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 2595 3851  2328 2619  2862 2619 2595 3851
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 3129 4341  2862 3319  3396 3319 3129 4341
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 3663 4635  3396 4019  3930 4019 3663 4635
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 4197 4733  3930 3669  4464 3669 4197 4733
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 4731 5069  4464 4369  4998 4369 4731 5069
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 5265 5139  4998 4719  5532 4719 5265 5139
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 5799 5461  5532 5069  6066 5069 5799 5461
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 6333 5811  6066 5419  6600 5419 6333 5811
2 3 0 1 -1 7 0 0 5 0.000 0 0 -1 0 0 4
	 6867 6091  6600 5769  7134 5769 6867 6091
2 1 0 1 -1 7 0 0 -1  0.000 0 0 -1 0 0 23
	1260 2928
	1794 2928
	1794 3250
	2328 3250
	2328 3851
	2862 3851
	2862 4341
	3396 4341
	3396 4635
	3930 4635
	3930 4733
	4464 4733
	4464 5069
	4998 5069
	4998 5139
	5532 5139
	5532 5461
	6066 5461
	6066 5811
	6600 5811
	6600 6091
	7134 6091
```

```
         7134 6119
2 1 1 1 -1 7 0 0 -1  4.000 0 0 -1 0 0 2
         1260 4362
         7134 4362
2 1 1 1 -1 7 0 0 -1  4.000 0 0 -1 0 0 2
         3396 4362
         3396 6119
```

B.4 Sample ddat .xg Output File

```
TitleText: zip1hires_day.xg: n=5

" Price (mean)
1 1.900571
2 1.934333
3 1.927000
4 1.954333
5 1.987000
6 1.994000
7 2.011667
8 2.012333
9 1.997333
10 1.990667
11 2.246000
12 2.217429
13 2.234286
14 2.243095
15 2.256286

" Price (-1s.d.)
1 1.764857
2 1.856941
3 1.880591
4 1.918859
5 1.949149
6 1.972693
7 2.000462
8 1.997973
9 1.969175
10 1.961961
11 2.213431
12 2.199214
13 2.222680
14 2.225546
15 2.242040

" Price (+1s.d.)
1 2.036285
2 2.011726
3 1.973409
4 1.989808
5 2.024851
6 2.015307
7 2.022872
8 2.026694
9 2.025492
10 2.019372
11 2.278569
12 2.235643
13 2.245891
14 2.260645
15 2.270531

" Alpha (mean)
1 12.181548
2 5.135457
3 4.203779
4 2.854103
5 1.958591
6 1.455158
7 1.127288
8 1.156317
9 1.611285
10 1.767713
11 2.942451
12 1.879179
13 1.092455
14 1.221152
15 1.390803

" Alpha (-1s.d.)
1 6.552933
2 1.666247
3 1.801715
4 1.052502
5 0.652610
6 1.008860
7 0.639893
8 0.635488
9 0.989116
10 1.090569
11 2.456126
12 1.083085
13 0.728010
14 0.792157
15 1.162194

" Alpha (+1s.d.)
1 17.810164
2 8.604667
```

```
3 6.605843
4 4.655703
5 3.264572
6 1.901456
7 1.614684
8 1.677147
9 2.233454
10 2.444857
11 3.428777
12 2.675273
13 1.456900
14 1.650147
15 1.619413

" Efficiency (mean)
1 98.000000
2 99.333333
3 100.000000
4 100.000000
5 100.000000
6 100.000000
7 100.000000
8 100.000000
9 100.000000
10 100.000000
11 100.000000
12 100.000000
13 100.000000
14 100.000000
15 100.000000

" Efficiency (-1s.d.)
1 95.333333
2 98.000000
3 100.000000
4 100.000000
5 100.000000
6 100.000000
7 100.000000
8 100.000000
9 100.000000
10 100.000000
11 100.000000
12 100.000000
13 100.000000
14 100.000000
15 100.000000

" Efficiency (+1s.d.)
1 100.666667
2 100.666667
3 100.000000
4 100.000000
5 100.000000
6 100.000000
7 100.000000
8 100.000000
9 100.000000
10 100.000000
11 100.000000
12 100.000000
13 100.000000
14 100.000000
15 100.000000

" Quantity (mean)
1 6.200000
2 6.000000
3 6.000000
4 6.000000
5 6.000000
6 6.000000
7 6.000000
8 6.000000
9 6.000000
10 6.000000
11 7.000000
12 7.000000
13 7.000000
14 6.800000
15 7.000000

" Quantity (-1s.d.)
1 5.800000
2 6.000000
3 6.000000
4 6.000000
5 6.000000
6 6.000000
7 6.000000
8 6.000000
9 6.000000
```

```
10 6.000000
11 7.000000
12 7.000000
13 7.000000
14 6.400000
15 7.000000

" Quantity (+1s.d.)
1 6.600000
2 6.000000
3 6.000000
4 6.000000
5 6.000000
6 6.000000
7 6.000000
8 6.000000
9 6.000000
10 6.000000
11 7.000000
12 7.000000
13 7.000000
14 7.200000
15 7.000000

" Dispersion (mean)
1 0.172134
2 0.073354
3 0.062094
4 0.042158
5 0.028930
6 0.021494
7 0.016651
8 0.017080
9 0.023800
10 0.026111
11 0.052813
12 0.033729
13 0.019608
14 0.021478
15 0.024963

" Dispersion (-1s.d.)
1 0.096840
2 0.024511
3 0.026613
4 0.015546
5 0.009640
6 0.014902
7 0.009452
8 0.009387
9 0.014610
10 0.016109
11 0.044084
12 0.019440
13 0.013067
14 0.014181
15 0.020860

" Dispersion (+1s.d.)
1 0.247427
2 0.122196
3 0.097575
4 0.068769
5 0.048221
6 0.028086
7 0.023850
8 0.024773
9 0.032990
10 0.036113
11 0.061542
12 0.048018
13 0.026150
14 0.028775
15 0.029067

" Volatility (mean)
1 0.257432
2 0.068596
3 0.041973
4 0.037325
5 0.025440
6 0.026026
7 0.016627
8 0.018294
9 0.023492
10 0.022558
11 0.069592
12 0.031745
13 0.018505
14 0.022640
15 0.022413
```

```
" Volatility (-1s.d.)
1 0.120674
2 0.021325
3 0.023222
4 0.021613
5 0.013213
6 0.018325
7 0.011129
8 0.012639
9 0.013094
10 0.016685
11 0.054013
12 0.023414
13 0.014316
14 0.012602
15 0.018563

" Volatility (+1s.d.)
1 0.394189
2 0.115867
3 0.060724
4 0.053036
5 0.037666
6 0.033727
7 0.022124
8 0.023950
9 0.033890
10 0.028430
11 0.085171
12 0.040076
13 0.022695
14 0.032678
15 0.026262
```

References

Agorics, Inc. (1996). Real-time video delivery with market-based resource allocation. Technical report ADd004P, Agorics, Inc., Los Altos, CA.

Agorics, Inc. (1997). Going, going, gone! A survey of auction types. `http://www.agorics.com/new.html` (URL correct 11 June 1997).

Arthur, W. B. (1993). On designing economic agents that behave like human agents. *Evolutionary Economics*, *3*, 1–22.

Axelrod, R. (1984). *The Evolution of Cooperation.* Basic Books, New York.

Baker, A. D. (1996). Metaphor or reality: A case study where agents bid with actual costs to schedule a factory. In Clearwater, S. H. (Ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pp. 184–223. World Scientific, Singapore.

Bannock, G., Baxter, R. E., & Davis, E. (1992). *The Penguin Dictionary of Economics* (fifth edition). Penguin, London.

Batali, J., & Kitcher, P. (1994). Evolutionary dynamics of altruistic behavior in optional and compulsory versions of the iterated prisoner's dilemma. In Brooks, R., & Maes, P. (Eds.), *Artificial Life IV*, pp. 343–348 Cambridge, MA. MIT Press Bradford Books.

Begg, D., Fischer, S., & Dornbusch, R. (1994). *Economics* (fourth edition). McGraw-Hill, London.

Bollerslev, T., & Domowitz, I. (1992). Some effects of restricting the electronic order book in an automated trade execution system. In Friedman, D., & Rust, J. (Eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, pp. 221–252. Addison-Wesley, New York.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology.* MIT Press Bradford Books, Cambridge MA.

Camerer, C. (1995). Individual decision making. In Kagel, J. H., & Roth, A. E. (Eds.), *The Handbook of Experimental Economics*, pp. 587–703. Princeton University Press, Princeton, NJ.

Cason, T., & Friedman, D. (1992). An empirical analysis of price formation in double auction markets. In Friedman, D., & Rust, J. (Eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, pp. 253–284. Addison-Wesley, New York.

Clark, A. J. (1997). *Being There: Putting Brain, Body, and World Together Again.* MIT Press Bradford Books, Cambridge MA.

Clearwater, S. H. (Ed.). (1996). *Market-Based Control: A Paradigm for Distributed Resource Allocation.* World Scientific, Singapore.

Clearwater, S. H., Costanza, R., Dixon, M., & Schroeder, B. (1996). Saving energy using market-based control. In Clearwater, S. H. (Ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pp. 253–273. World Scientific, Singapore.

Cliff, D., & Noble, J. (1997). Knowledge-based vision and simple visual machines. *Philosophical Transactions of the Royal Society of London: B*. In Press: Also available as University of Sussex School of Cognitive and Computing Sciences Technical Report CSRP459.

Davis, D. D., & Holt, C. A. (1993). *Experimental Economics*. Princeton University Press, Princeton, NJ.

de la Maza, M., & Yuret, D. (1994). A futures market simulation with non-rational participants. In Brooks, R. A., & Maes, P. (Eds.), *Artificial Life IV*, pp. 325–330. MIT Press Bradford Books, Cambridge MA.

Drexler, K. E., & Miller, M. S. (1988). Incentive engineering for computational resource management. In Huberman, B. A. (Ed.), *The Ecology of Computation*, pp. 231–266. Elsevier/North-Holland.

Easley, D., & Ledyard, J. (1992). Theories of price formation and exchange in double oral auctions. In Friedman, D., & Rust, J. (Eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, pp. 63–98. Addison-Wesley, New York.

Friedman, D. (1992). The double auction market institution: A survey. In Friedman, D., & Rust, J. (Eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, pp. 3–26. Addison-Wesley, New York.

Friedman, D., & Rust, J. (Eds.). (1992). *The Double Auction Market: Institutions, Theories, and Evidence*. Addison-Wesley, New York.

Fudenberg, D., & Tirole, J. (1991). *Game Theory*. MIT Press, Cambridge, MA.

Glance, N. S., & Huberman, B. A. (1992). Dynamics with expectations. *Physics Letters A*, *165*, 432–440.

Glance, N. S., & Huberman, B. A. (1994). The dynamics of social dilemmas. *Scientific American*, *270*(3), 76–81.

Gode, D. K., & Sunder, S. (1992). Lower bounds for efficiency of surplus extraction in double auctions. In Friedman, D., & Rust, J. (Eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, pp. 199–219. Addison-Wesley, New York.

Gode, D. K., & Sunder, S. (1993). Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, *101*(1), 119–137.

Gravelle, H., & Rees, R. (1992). *Microeconomics* (second edition). Longman, London.

Harty, K., & Cheriton, D. (1996). A market approach to operating system memory allocation. In Clearwater, S. H. (Ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pp. 126–155. World Scientific, Singapore.

Heap, S. P. H., & Varoufakis, Y. (1995). *Game Theory: A Critical Introduction*. Routledge, London.

Hillis, W. D. (1985). *The Connection Machine*. MIT Press, Cambridge MA.

Hodgson, G. M. (1993). *Economics and Evolution: Bringing life back into economics.* Polity Press, Cambridge.

Holt, C. A. (1995). Industrial organization: A survey of laboratory research. In Kagel, J. H., & Roth, A. E. (Eds.), *The Handbook of Experimental Economics*, pp. 349–443. Princeton University Press, Princeton, NJ.

Huberman, B. A. (Ed.). (1988a). *The Ecology of Computation.* Elsevier/North-Holland.

Huberman, B. A. (1988b). The ecology of computation. In Huberman, B. A. (Ed.), *The Ecology of Computation*, pp. 1–4. Elsevier/North-Holland.

Kagel, J. H. (1995). Auctions: A survey of experimental research. In Kagel, J. H., & Roth, A. E. (Eds.), *The Handbook of Experimental Economics*, pp. 501–585. Princeton University Press, Princeton, NJ.

Kagel, J. H., & Roth, A. E. (Eds.). (1995). *The Handbook of Experimental Economics.* Princeton University Press, Princeton, NJ.

Kagel, J. H., & Vogt, W. (1992). Buyer's bid double auctions: Preliminary experimental results. In Friedman, D., & Rust, J. (Eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, pp. 285–306. Addison-Wesley, New York.

Marron, D. B., & Bartels, C. W. (1996). The use of computer-assisted auctions for allocating tradeable pollution permits. In Clearwater, S. H. (Ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pp. 274–299. World Scientific, Singapore.

May, R. M., Bohoeffer, S., & Nowak, M. A. (1995). Spatial games and the evolution of cooperation. In Morán, F., Moreno, A., Merelo, J. J., & P.Chacón (Eds.), *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life (ECAL95)*, pp. 749–759 Berlin. Springer-Verlag.

Miller, M. S., & Drexler, K. E. (1988a). Comparative ecology: A computational perspective. In Huberman, B. A. (Ed.), *The Ecology of Computation*, pp. 51–76. Elsevier/North-Holland.

Miller, M. S., & Drexler, K. E. (1988b). Markets and computation: Agoric open systems. In Huberman, B. A. (Ed.), *The Ecology of Computation*, pp. 133–176. Elsevier/North-Holland.

Miller, M. S., Krieger, D., Hardy, N., Hibbert, C., & Tribble, E. D. (1996). An automated auction in ATM network bandwidth. In Clearwater, S. H. (Ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pp. 96–125. World Scientific, Singapore.

Nottola, C., Leroy, F., & Davalo, F. (1991). Dynamics of artificial markets: Speculative markets and emerging 'common sense' knowledge. In Varela, F. J., & Bourgine, P. (Eds.), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life (ECAL91)*, pp. 185–194. MIT Press Bradford Books, Cambridge, MA.

Roth, A. E. (1995). Introduction to experimental economics. In Kagel, J. H., & Roth, A. E. (Eds.), *The Handbook of Experimental Economics*, pp. 3–109. Princeton University Press, Princeton, NJ.

Roth, A. E., & Oliveira Sotomayor, M. A. (1990). *Two-Sided Matching: A study in game-theoretic modeling and analysis.* Cambridge University Press, Cambridge.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., & McClelland, J. L. (Eds.), *Parallel Distributed Processing, Volume 1: Foundations*, pp. 318–362. MIT Press Bradford Books, Cambridge MA.

Rust, J., Miller, J., & Palmer, R. (1992). Behavior of trading automota in a computerized double auction market. In Friedman, D., & Rust, J. (Eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, pp. 155–198. Addison-Wesley, New York.

Satterthwaite, M., & Williams, S. (1992). The bayesian theory of the k-double auction. In Friedman, D., & Rust, J. (Eds.), *The Double Auction Market: Institutions, Theories, and Evidence*, pp. 99–124. Addison-Wesley, New York.

Smith, V. L. (1962). An experimental study of competitive market behavior. *Journal of Political Economy*, *70*, 111–137.

Smith, V. L. (1992). *Papers in Experimental Economics.* Cambridge University Press, Cambridge.

Smith, V. L., & Williams, A. (1992). Experimental market economics. *Scientific American*, *267*(6), 72–77.

Stanley, E. A., Ashlock, D., & Tesfatsion, L. (1993). Iterated prisoner's dilemma with choice and refusal of partners. In Langton, C. G. (Ed.), *Artificial Life III: Proceedings of the workshop on Artificial Life, June 1992*, Vol. XVII of *Sante Fe Institute Studies in the Sciences of Complexity*, pp. 131–176 Reading, MA, USA. Addison-Wesley.

Sunder, S. (1995). Experimental asset markets: A survey. In Kagel, J. H., & Roth, A. E. (Eds.), *The Handbook of Experimental Economics*, pp. 445–500. Princeton University Press, Princeton, NJ.

Tesfatsion, L. (1997). How economists can get Alife. In Arthur, W. B., Durlauf, S., & Lane, D. (Eds.), *The Economy as a Complex Evolving System II.* Addison Wesley.

Thomsen, J. S., Mosekilde, E., & Sterman, J. D. (1992). Hyperchaotic phenomena in dynamic decision making. *Journal of Systems Analysis and Modeling Simulation (SAMS)*, *9*, 137–156.

Vromen, J. J. (1995). *Economic Evolution.* Routledge, London.

Wilmott, P., Howison, S., & Dewynne, J. (1995). *The Mathematics of Financial Derivatives.* Cambridge University Press, Cambridge.

Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation*, *2*(1), 1–18.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, *3*(2), 149–175.