

**A Simulation of a Simple Stock Market
Environment using Python**

Charlotte Lucy Fitton

Computing for Business BSc
Session 2008/2009

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)_____

Summary

In a world where money is so often a simulated concept, it is important to be able to accurately re-create the transaction processes that surround daily life in order to gain understanding and to predict potential issues. The problem with this however, is that it is impossible for a computer to display the same level of cognitive reasoning and intelligence that a human presents and as such attempts to replicate behaviour are extremely complex. There has been much research into how humans react in a static stock-market type environment within which they quickly perform to a state where each person has maximised their advantage known as an equilibrium. Movement to equilibrium, known as convergence, by humans has been proved to be highly efficient and it has been attempted to emanulate these results by using computer programs.

The programs that give the best results work in one of two ways; they either implement Artificial Intelligence or an influencing Market Structure.

This project intends to explore the human, AI and Market Structure results in parallel with economic theories, patterns and behaviour to evaluate whether human intelligence can indeed be matched and design a program which attempts to do so.

Acknowledgements

First and foremost I would like to thank my supervisor Dr Marc DeKamps for his continuous support, encouragement and guidance throughout the year, this project would not exist without it. Thanks too to Dr. Haiko Müller for his feedback on the mid-project report and the project demonstration and to Dr Dan Ladley whose PhD inspired my choice of projects and for his excellent suggestion for research. Also to the School of Computing for letting me stay year after year and for the assistance provided by it's wonderful staff, especially Dr Mark Walkey who appears to have a bottomless pit of patience and optimism.

I am eternally grateful to my fellow course mates for their companionship in Dec-10 until the early hours, particularly Matt Revell for always spotting that missing semicolon, you are my syntax saviour. To all my house mates on both sides of the road, for being experiment guinea pigs, providing hugs and educating me on the finer points of Rugby League and the dietary habits of lions.

And finally, but by no means least, my family and friends for their unstinting belief that I would eventually graduate and Eleanor, for absolutely everything.

Contents

1	Project Introduction	1
1.1	Problem Statement & Motivation	1
1.2	Aims	2
1.3	Objectives	2
1.4	Minimum Requirements	2
1.5	Relevance to Degree	3
1.6	Project Management	3
2	Background Reading & Research	4
2.1	Introduction	4
2.2	Economic Background	5
2.2.1	Supply & Demand	5
2.2.1.1	Volume Tunnels & Price Tunnels	6
2.2.2	Market-Based Control	8
2.2.3	Walras Tâtonnement	8
2.2.4	Co-efficient of Convergence	8
2.2.5	Allocation Efficiency	9
2.2.6	Profit Dispersion	9
2.2.7	Market Organisation	9
2.3	Agent Behaviour Background	10
2.3.1	Stigmergy	10
2.3.2	Swarm Intelligence and Emergence	10
2.4	Background Reading	11

2.4.1	Cliff and Bruten	11
2.4.2	Vernon Smith	12
2.4.3	Gode and Saunder	15
2.5	Conclusion	18
3	Methodology	20
3.1	Introduction	20
3.2	The Waterfall Model	20
3.3	The Spiral Model	21
3.4	Conclusion	21
4	Design & Implementation	23
4.1	Introduction	23
4.2	Design Considerations	23
4.3	Software Justification	24
4.4	System Architecture	26
4.5	Mathematical Design	26
4.5.1	Transaction Algorithm Design	26
4.5.2	Bounded Transaction Algorithm Design	28
4.6	Iteration 1	29
4.7	Iteration 2	29
4.8	Iteration 3	30
4.9	Iteration 4	32
4.10	Iteration 5	33
5	Testing Design	35
5.1	Introduction	35
5.2	Minimum Criteria Testing	35
5.2.1	Test One	35
5.2.2	Tests Two & Three	35
5.2.3	Test Four	36
5.3	Extension Tests	36

5.4	Human Trader Tests	36
5.4.1	Smith Tests	36
6	Results	37
6.1	Minimum Criteria Results	37
6.1.1	Test One	37
6.1.2	Tests Two & Three	38
6.1.3	Test Four	38
6.2	Test Five	40
6.3	Human Trader Tests	40
7	Evaluation	44
7.1	Introduction	44
7.2	Overall Evaluation of Program	44
7.2.1	Criteria	45
7.2.2	Further Development	45
7.3	Possible Improvements	46
7.4	Project Report	46
7.5	Methodology	47
7.6	Solving the Problem Scenario	47
7.7	Human vs ZI vs ZIB	48
	Bibliography	49
A	Personal Reflection	51
B	Appendix B -Test results	54
B.1	Introduction	54
B.2	Minimum Criteria Results	54
B.2.1	Test One- Extras	54
B.2.2	Tests Four and Five- Extras	54
C	Abbreviation Reference List	59

Chapter 1

Project Introduction

1.1 Problem Statement & Motivation

The allocation and price of resources is a topic which influences every aspect of our lives from what is eaten for breakfast to the quality of education available to staff wages and in the current economic climate, it has become increasingly important to understand how to maximise utilisation of these resources in order to benefit the greatest number of people. The stock market is often indicative of how resources are distributed and how prices are agreed as, for all ‘stock’, there is massive scope for buying and bargaining. This system is currently controlled by a number of traders who buy and sell units according to cost and demand. This is a very well-paid profession as it is assumed that to optimise allocative efficiency requires highly-skilled and knowledgeable persons, yet experiments have shown that when simulation is applied, using only simple rules and with no market knowledge, both artificial and human ‘Zero-Intelligence’ agents will converge to an equilibrium where, not only is allocation efficiency high, but the percentage of transactions occurring is also great.

Many research papers have been written with different hypotheses on why this occurs, some suggest it is the market structure, others, the intelligence of the agents. This report aims to discover how, by evaluating past research and applying a combination of price theory methods onto a Zero-Intelligence model, the ultimate transaction price is reached. This will be done by re-creating experimental scenarios as outlined in relevant papers and run with a bespoke program created to test a hypothesis. This will produce results that can be compared and contrasted and allow for exploration of respective conclusions for evaluation.

This report aims to show that the consequence of attempting these experiments will prove that

even ‘Zero-Intelligence’ agents converge to equilibrium when simple rules are applied and that one additional factor will prove itself to be the most instrumental to convergence. The factor that will be implemented within this bespoke program combines the two main theories to create an intelligent market structure. This works by implementing bounds based on past bids and transactions hence creates a learning environment.

1.2 Aims

To create and evaluate a simulation of a simple static stock-market environment containing trader agents. This will be done by using arrays of Zero-Intelligence buyer and seller agents which are each assigned a quantity of stock to transfer in an allocated time period. During this time period, randomly-generated bids and offers created by these agents will be observed and matched in a transaction array, these transactions will be mapped in a simple Graphical User Interface, GUI, which, when viewed over many time periods, will display evidence to what extent, if any, there is convergence to the theoretical market equilibrium.

1.3 Objectives

- Create an abstract world with buyer and seller agents capable of generating random numbers within constraints
- Allow capabilities for transactions to take place
- Display past transactions graphically
- Observe if transaction price tends towards the predicted equilibrium and how long this takes

1.4 Minimum Requirements

- Creation of ‘Buyer’ and ‘Seller’ arrays
- Creation of ‘Transaction Dictionary’
- Transaction matching algorithm
- GUI capable of displaying past transactions

1.5 Relevance to Degree

This project is appropriate to Computing for Business as it draws upon many modules within the degree program. This includes the programming modules SE12, SE15, SE20 & SE24, the artificial intelligence modules AI21 & AI22 and maths modules MA11 & MA12. The following modules are also extremely useful for background knowledge; AI23-Bio-inspired Computing which teaches many of the AI concepts covered in background reading and INMA2-Modelling and Simulation which is a business maths module which assists with mathematical modelling. The use of computer simulation to model a common business practice makes this project degree appropriate.

1.6 Project Management

This project was carried out over two semesters which required organising basic time management early on. It was decided to spend the first semester focusing mainly on background reading & research which resulted in a mid-term report. Contained in this report was a timetable which outlined deliverables to be performed in the second semester. This included design, implementation and evaluation. This report contains all work performed and the original project schedule is in Appendix D

The progress of the report and success of the deliverables were discussed in weekly supervisor meetings which became bi-weekly around 2 months from the final deadline.

Chapter 2

Background Reading & Research

2.1 An Introduction to Background

Since the first book was published on modern economics [17], there have been countless published theories attempting to predict future trends, behaviours and markets. To completely understand this field is beyond the scope of this degree program, but here it shall be attempted to cover a relevant cross-section of topics and concepts in order to fully explore the project.

As a project which is heavily dependant on research already preformed, this chapter is integral to understanding the problem situation and shall concentrate primarily on the 1993 Cliff paper [5] which can be considered a fairly comprehensive template upon which to base the program situation. Cliff makes particular reference to two other papers, Smith[18] and Gode [7] hence these will be investigated thoroughly. There are several other sources which are also appropriate to the subject matter, these shall be briefly detailed where features are considered to be congruent or could enhance understanding. To observe differing experiments and their results could aid in design and evaluation of the project.

The books and papers cited in this project [5],[7],[18],[6],[11] preform variations of simple, typical experiments as described in Chapter 4 ‘Experimental Economics’ of the ‘Cliff’ paper. A number of controlled market-based environments were constructed with groups of human subjects and/or simple ZI agents. The basic parameters are similar to those that will be preformed in the program in several aspects. Simple upper and lower bounds of possible bids were introduced, each trader was assigned a hypothetical value for stock, creating a volume tunnel, see section 2.4.3, of ‘costs’, the format is that of an open-outcry double auction trading pit, there were no cost incurred through bids or trading, there are “days” or distinct trading periods and strict privacy regarding knowledge of the market is maintained.

Unless otherwise stated, assume the experiments outlined have these factors.

Despite being primarily a Computing project, there are some basic economic ideas and other models of agent behaviour which shall also be outlined.

2.2 Economic Background

2.2.1 Supply & Demand

In its most simple state, “supply and demand” dictates price determination. When the desired quantity of a resource to be bought is higher than the quantity that can be supplied, the price will rise due to the suppliers being able to command the highest price available for their goods, this is known as a “sellers market”. This has two results; some consumers will be unable or unwilling to afford the resource leading to a drop in quantity demanded and the quantity supplied may increase as suppliers become more willing to sell at the inflated price. Similarly when there is a decrease in demand, the price of the resource will drop to allow more consumers the ability to purchase and may decrease the number of suppliers, known as a “buyers market”. This results in the graph 2.1.

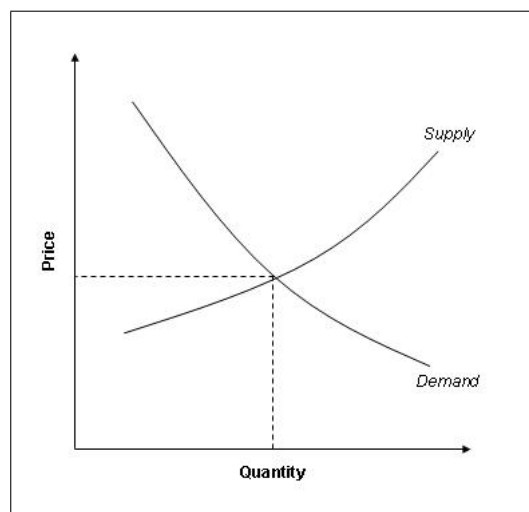


Figure 2.1: Supply and Demand Curves.

Basically, excess demand creates increased prices whereas excess supply means decreased prices[20]. Where demand and supply lines cross this is an equilibrium value. This is where quantity demanded matches quantity supplied at a specified time and price. According to classical economic theories, if supply and demand are fixed, there is always a theoretical equilibrium value and the market will “self-

correct” until it reaches that stabilised state where neither buyer nor seller have incentive to change their prices.

Smith[?] also referred to ‘supply and demand’ as ‘market supply and demand’ because due to the privacy that restricted trader knowledge coupled with their motivation to maximise profit, the apparent supply and demand schedules could be rather different to the actual, underlying trade abilities. Inexperienced human traders in an open-outcry continuous double action, detailed in 2.2.7, will make an initial guess then, as other traders bid, they modify these guesses appropriately causing the market to ‘shift’ dynamically as trades are made and traders leave.

2.2.1.1 Volume Tunnel & Price Tunnels

Volume Tunnels & Price Tunnels are tools which allow for closer observation of the Supply and Demand curve model, this is because the curves depict a simplistic and generalised view of market[12]. ‘Most classical theories of price determination and equilibrium 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’[18]. However, where there are few traders in the market, the ‘curves’ will appear step-like as in 2.2. This is important to this project as the program will have small numbers of traders to allow clear distinction of behaviour. All bids, offers and transactions that occur within the volume tunnel represent non-loss making behaviour.

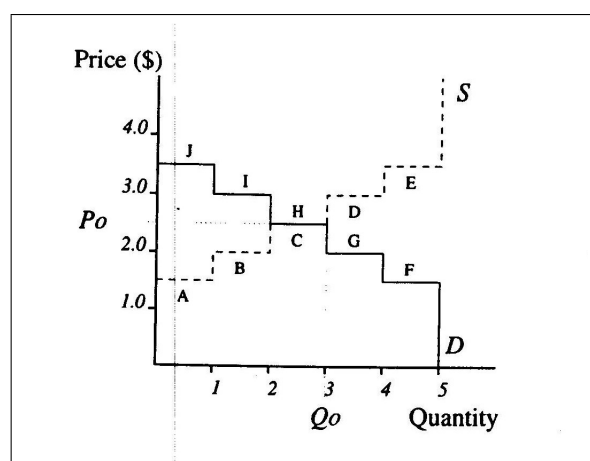


Figure 2.2: A Supply and Demand Graph in a market where there are 5 buyers and 5 sellers. Image courtesy of[5]

Figure 2.2 represents an example described in Cliff [5] where five sellers, represented by the letters

A to E and five buyers, F to J are assigned a limit price(in dollars) in regards to trade one piece of stock. The limit prices are as follows:

- Sellers: A= \$1.50, B=\$2.00, C = \$2.50, D = \$3.00, E = \$3.50
- Buyers: F= \$1.50, G=\$2.00, H = \$2.50, I = \$3.00, J = \$3.50

Where traders are allowed to enter into deals that do not maximise their profit-making ability, known as marginal deals due to the small margin of profit, the clearing quantity, which is the amount of items sold, will be three. This is because the equilibrium price pictured is \$2.50 which allows Sellers A-C to trade as well as Buyers F-H. This ‘horizontal overlap of supply and demand’ is referred to as a “Volume Tunnel” [6].

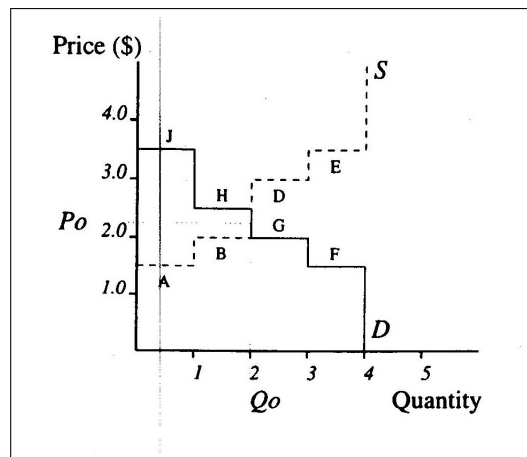


Figure 2.3: A Price Tunnel in a market where there are 4 buyers and 4 sellers. Image courtesy of[5]

However, transactions happen individually and not necessarily in an optimal fashion, so where 2 traders are matched, as in 2.3 where seller C and buyer I have made a transaction and hence been removed from the market, the Supply and Demand graph looks markedly different. The clearing quantity has been reduced and the equilibrium price is no longer a 'scalar value: rather there is a bounded range of possible equilibrium values'[5] meaning at any point between the values \$2-2.50, the resulting trade possibilities are the same; 2 trades can occur and the allocation efficiency will be identical. This vertical range of equilibrium values is known as “Price Tunnel”

2.2.2 Market-Based Control

The research field of Market-Based Control, MBC, views economics as a model for other applications, 'economics can act as a valuable source of terminology, inspiration, and metaphors for developing solutions to problems in distributed resource-allocation and control.' [5]

By simulating a market-like virtual environment, software-activated economic agents will apply distributed decision mechanisms which display patterns which not only are surprisingly co-ordinated, section 2.3, but are applicable to model several actual common activities. Examples suggest by Cliff[5] include allocation of network bandwidth, air conditioning and pollution regulation.

He also claims that currently no MBCs are both automated and decentralised. They either 'devolve power from human operators' and hence are reliant on the accuracy of humans, or use 'centralised 'auctioneer' processes' and that to be able to do both would minimise reliance on central control mechanisms meaning that the system would be stronger and independent of any singular system. This could be a useful addition to any system, common or otherwise.

2.2.3 Walras Tâtonnement

Leon Walras [20] suggested that in a market with perfect competition, equilibrium can be attained regardless of original disequilibrium position or convergence route, through a process of 'groping'.

Buyers and sellers make known their prices in an initial state of bidding. In the second iteration, published prices are increased where there is excess demand and reduced when shortfall occurs. The process continues until there is a balance of supply and demand in all markets.

2.2.4 Co-efficient of Convergence

The co-efficient of convergence, α , can be used to quantify the degree to which transactions move towards the equilibrium at the end of each day trading. It can be computed in the formula

$$\alpha = 100\sigma_0/P_0$$

where " σ_0 is the standard deviation of exchange prices around the equilibrium rather than the mean exchange price"[19] Smith's experiments show decline in the value of σ as trading days progress, this is because the amount of convergence is less due to the increasing accuracy of the opening trade values.

This is applicable where a set of transaction prices can be denoted by:

$$P_i : i \in \{1, \dots, n\}; \sigma_0 = \sqrt{\frac{1}{n} \sum_{i=1}^n (P_i - P_0)^2}$$

2.2.5 Allocation Efficiency

Defined as 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. If equilibrium is reached by all traders, allocation efficiency will be at 100% .This can be used as a measure of intelligence and efficiency

2.2.6 Profit Dispersion

Defined as by Cliff[5] 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 the units are traded at the equilibrium price.’ When a_i is the actual profit earned by trader i, and p_i is the theoretical equilibrium profit for trader i, for group of traders, size n, profit dispersion is shown as

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (a_i - p_i)^2}$$

It is a measure of variance in profit and the lower the result, the higher the intelligence and efficiency displayed. Gode [7]states that decline in dispersion ‘suggest that, in contrast to aggregate learning, distributional aspects of market performance may be sensitive to human motivation and learning.’

2.2.7 Market Organisation

The type of auction that occurs in the majority of experiments reviewed and to an extent that the program shall utilise is an open-outcry continuous double action. It has information arriving in a continuous asynchronous stream[5] by both buyers and sellers in no particular order. However, it may be argued that if for example two buyer ‘shout’ the same amount, priority will be given to the first.

A double auction is a ‘multi-lateral process in which buyers as well as sellers can freely enter limit orders(bids or asks) and accept asks or bids entered by others’[7]

Walrasian auctions follow walras tâtonnement, section 2.2.3, and submit bids to a central auctioneer who will only allow transactions to take place at equilibrium prices, hence creating perfect information.

Exchange markets allow agents to ‘circulate and engage in bilateral higgling[sic] and bargaining until they make a contract or the trading period ends’,[11],[18], it has been argued that due to containing the bargaining behaviour between few traders as opposed to the whole market, agents do not display learning. This would be extremely difficult to implement and is far beyond the scope of this project.

Perfect competition is a theoretical market structure which occurs when singular agents have no

influence over the market as they are too small a part, all traders seek to maximise profit, all traders can enter and leave the market freely, perfect or complete information is available and all goods are interchangeable or homogeneous[12].

2.3 Agent Behaviour Background

2.3.1 Stigmergy

Stigmergy was defined by Grassè as ‘stimulation of workers by the performance they have achieved’[4], by leaving a “trace” of an action the environment will be altered and spontaneous, indirect coordination between subsequent agents or actions can occur because of reaction to this trace. Although Stigmergy is usually used to refer to the behaviour of social insects, it is applicable in an economic sense in that the action of every agent modifies the microeconomic environment slightly. Over a significant period of time this, in turn, will alter perceptions and possibly the actions of ensuing agents. This is known as “the stigmergic phenomenon”.

2.3.2 Swarm Intelligence and Emergence

These two properties are very closely linked, especially in how they relate to the report and program. Almost an extension of stigmergy, swarm intelligence refers to the simulation of intelligence, artificial or instinctive, in the collective behaviour of agents with some degree of awareness. Simple rules are implemented and the individual agents obey. The, seemingly random, interactions between these agents become increasingly co-ordinated through stigmergy 2.3.1 and lead to the emergence of “intelligent” global behaviour without the use of a central control structure.

Emergence is the way behaviours or structures literally emerge from real-life applications of swarm or group intelligence. Often decentralised systems which have many agents each performing simple interactions create, when viewed as a collective, complex systems or patterns. The stock market portrays emergence in that without a singular central control structure, traders “follow” each other in creating share-prices, especially in extreme booms or busts. Emergence is sometimes known as competitive equilibrium [5].

2.4 Background Reading

2.4.1 Cliff and Bruten

This project is heavily influenced by the paper ‘Minimal-Intelligence Agents for Bargaining Behaviours in Market-based Environments’[5]. Describing how bargaining behaviours ‘converge to theoretical equilibrium price’ over time, they state that assigning differing levels of intelligence to buyer and seller agents affects rate and pattern. This is demonstrated by creating Zero Intelligence Agents (ZI) and Zero Intelligence Plus Agents (ZIP) to compare their results. A ZI trader is defined as having ‘no intelligence, does not seek or maximise profits, and does not observe, remember or learn’[7] throughout this paper.

The paper begins by describing economic movement and its relevance to the field of Computing. Asserting that ‘economics can act as a valuable source of terminology, inspiration and metaphors for developing solutions to problems in distributed resource-allocation and control’ leads to the domain of ‘Market Based Control’,MBC. This divides the scarce resources into units of commodity, or “stock” as it will be referred to in this report, to be traded for units of currency by buyers and sellers ‘within a market-like framework’.

Chapter 2 describes the economic theories, such as economy of supply and demand, walras tâtonnement and emergence, these are all detailed later in this chapter.

Chapter 6 contains information of how Cliff himself viewed the market and performed his experiments. It also describes how a Zero-Intelligence Plus, ZIP, trader has been created that adjust their profit margins using market price information. It is done by considering four factors, the first of which is whether it is active or inactive, whether it has remaining stock. If it does and is active, The three remaining factors all regard the latest bid: ‘its price, q ; whether it was a bid or an offer;and whether it was accepted or rejected. [...]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 trader should avoid lowering profit margin and only raise it if an acceptance of q suggests higher profits are available. ‘At any given time, t , a ZIP trader, i , can calculate their shout price $p_i(t)$ for unit j with limit price λ_{ij} using the traders real-valued profit-margin $\mu_i(t)$ according to the following equation:

$$p_i(t) = \lambda_{ij}(1 + \mu_i(t))$$

This calculation means once equilibrium is reached, there is ‘very little variance’ because of how profit margin will only change if it is affected by outside influence and as that is the factor which most

influences price change arriving at equilibrium begets a cycle of uniform trading. The profit margin is also designed to assist reaching equilibrium initially as seen in figure 2.4.

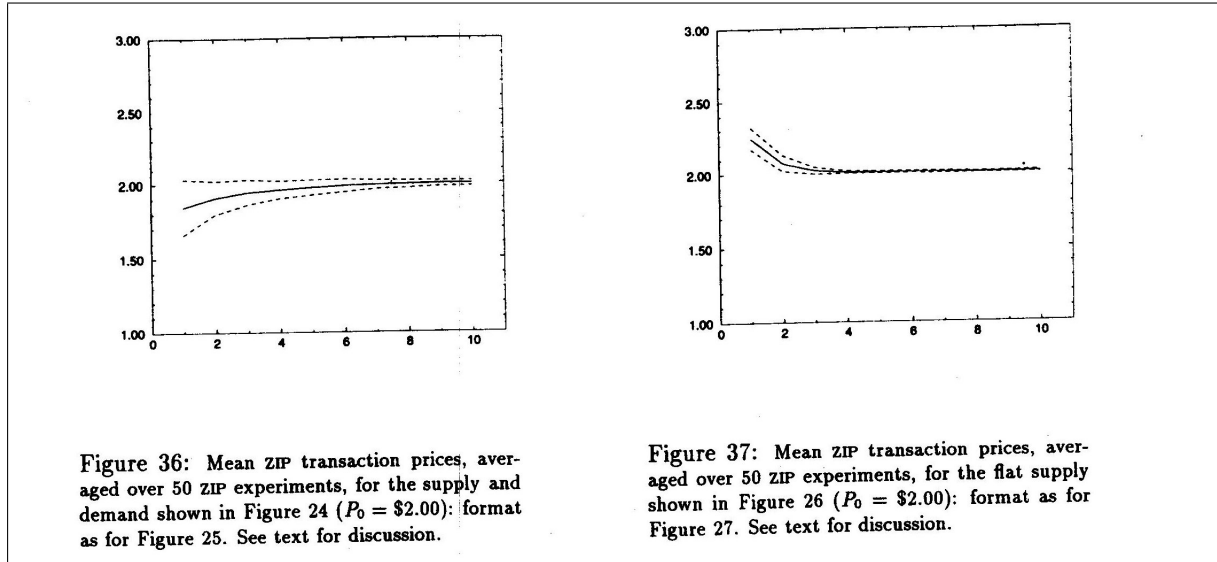


Figure 2.4: Cliff's experiments using ZIP traders. Mid-line is the mean value(of 50 experiments), upper line is mean plus one deviation, lower line is mean minus one deviation. Image courtesy of[5]

Results reveal high allocative efficiency, often 100% and profit dispersion falls steeply then remains consistently low which displays high intelligence. 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[Zero- Intelligence Controlled] traders.' This shows that Cliff's hypothesis that convergence to equilibrium is indeed determined more by intelligence than market structure. The results were also compared to Smith's human traders and found much of the behaviour exhibited in the various experiments matched which suggest human reasoning has been successfully simulated.

2.4.2 Vernon Smith

The first to perform 'Experimental Economics', the 1962 paper 'An experimental study of competitive market behaviour' [18] was written on the results of six years of human trading regimes which began by strictly adhering to the preliminary parameters as previously stated, this often gave graphs similar to

Variations were introduced to prove his hypothesis by induction, most notably Smith experimented with asynchronous supply and demand to test his theory that different ratios 'affected the nature of the approach of transaction prices to the theoretical equilibrium price'. The results show that where

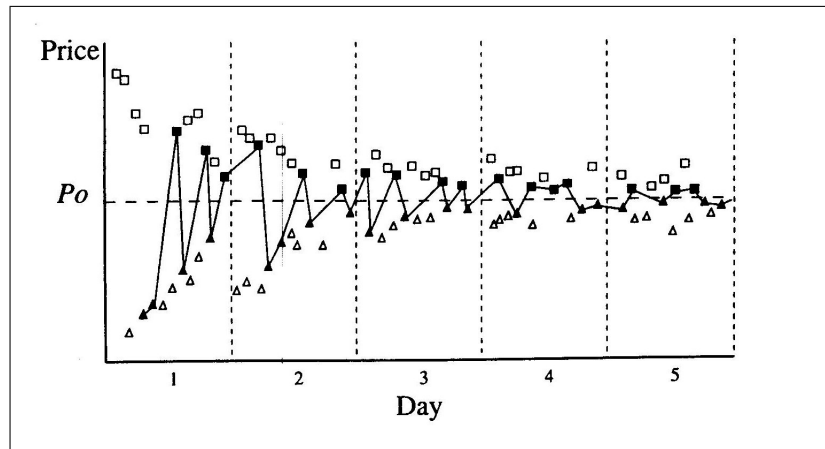


Figure 2.5: Synthetic data showing an idealised 5-day market experiment. Bid are shown as triangles, offers are shown as squares. Successful transactions are filled symbols and linked. Image courtesy of[5]

supply is ‘perfectly elastic’, displayed in figure 2.6, and appears flat, aligned with the equilibrium value, transaction prices ‘approach from above, but level out [slightly] above equilibrium’. Critically speaking, this result is interesting as the human bidders surely would implement some learning intelligence, yet the buyers were prepared to pay more than the equilibrium value for stock. This maybe could be considered an example of stigmergy, section 2.3.1.

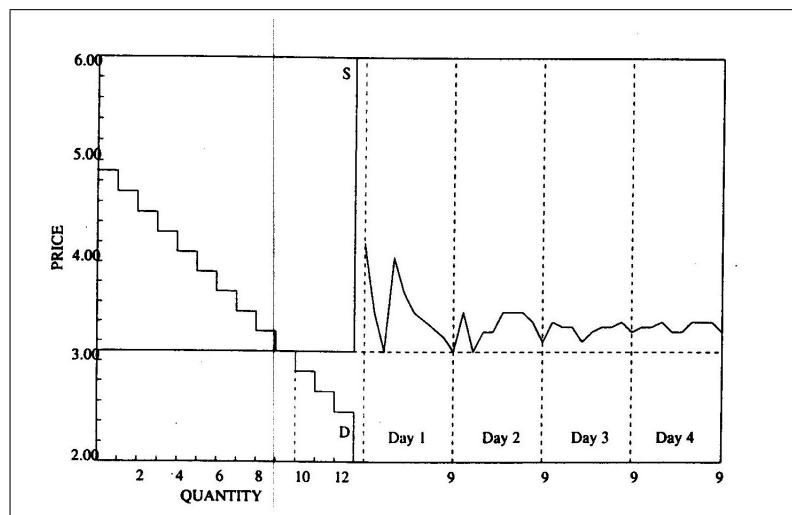


Figure 2.6: ‘Perfect Elastic’ Supply- Adapted from Smith’s(1962) Chart 4. Image courtesy of[5]

Smith also discovered that excess demand results in transaction approaching equilibrium ‘very slowly and from below.’ as seen in figure 2.7. This could be argued that although the economics of

supply and demand, see section 2.2.1, suggest that excess demand should produce initially high prices, possibly a single person has suggested a low ‘start’ price which has been followed by other traders, suggesting stigmergy, section 2.3.1 which evolves using tâtonnement, section 2.2.3, or perhaps swarm intelligence and emergence, section 2.3.2. However the human subject are influenced, equilibrium convergence takes three days.

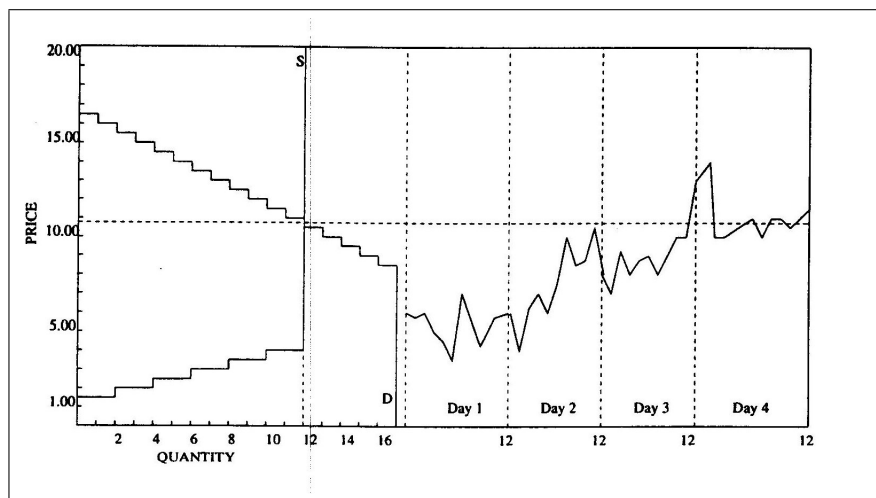


Figure 2.7: Excess Demand- Adapted from Smith's(1962) Chart 6. Image courtesy of[5]

During another experiment, buyers were made to stay silent during bidding except to accept offers to recreate a retail market, which is pictured in graph 2.8. The behaviour exhibited here is claimed to be because ‘since only sellers were making offers, the prices initially tended to be very much above equilibrium. Five of these offers were accepted at prices ranging from \$2.69 to \$2.80’, competition then caused prices to fall leading those who had paid the inflated price originally to abstain from entering into highly priced transactions which kept prices from converging to equilibrium.

The constraint of only being allowed to buy or sell a single unit was relaxed after the first eight experiments, however this is not explored in the Cliff paper nor will be re-created in this project

As can be seen in the experiments preformed by this paper, there is always evidence of movement towards equilibrium, despite the constraints imposed. This convergence shows intelligence in human traders and strengthens the hypothesis made by both Cliff and this paper that speed of convergence is dependant on learning behaviour in agents. It served to prove that ‘with remarkably little learning, strict privacy, and a modest number, inexperienced traders converge rapidly to a competitive equilibrium [...] Economic Agents do not need to have perfect knowledge of supply and demand.’[18]

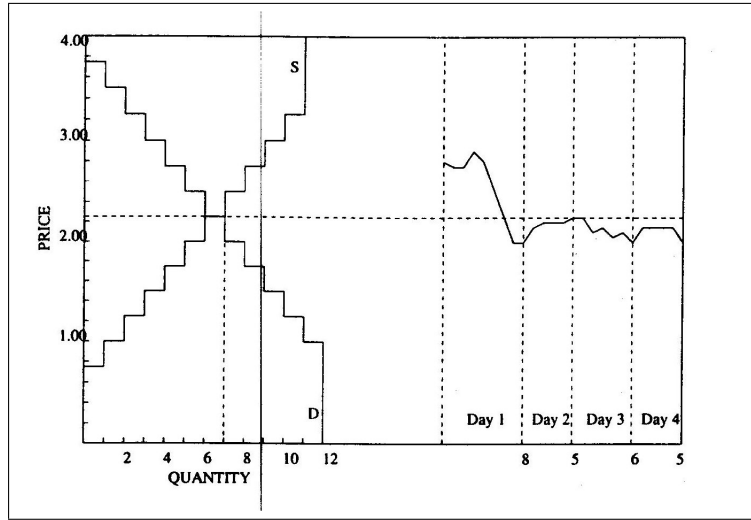


Figure 2.8: Passive Buyers- Adapted from Smith's(1962) Chart 8. Image courtesy of[5]

The findings made were ground breaking in their day and created the terms 'Co-efficient of Convergence', described in section 2.2.4 & 'Allocation Efficiency', section 2.2.5.

2.4.3 Gode and Saunder

Written before the Cliff[5] paper, and heavily influencing it, the Gode paper[7], 'Allocative Efficiency of Markets with Zero-Intelligence Traders:Market as a Partial Substitute for Individual Rationality' contains many of the same economic points and claims, but with one major difference, it is stated that;

'Imposing a budget constraint(i.e., not permitting traders to sell below their costs or buy above their values) is sufficient to raise the allocative efficiency of these auctions close to 100%. Allocative efficiency of a double auction derives largely from its structure, independent of traders motivation, intelligence or learning.'This claims that market structure is the most integral feature of convergence of non-loss making transactions toward equilibrium and have attempted to prove this hypothesis using primarily computer simulated traders.

This is done by first creating a market with both human traders, who were business graduates, and ZI traders who were to generate random bids or offers of integers between 1-200 with probability, P , of bid, i , or offer, j , as

$$P_i = 1/200; i = 1, 2, \dots, 200 \text{ or } P_j = 1/200; j = 1, 2, \dots, 200$$

A double auction was used in which 'each bid, ask, and transaction was valid for a single unit. A transaction cancelled[sic] any unaccepted bids and offers. Finally, when a bid and ask crossed, the

transaction price was equal to the earlier of the two’.

The experiment ran with three types of traders in simultaneous market. The first market contained the unconstrained traders, creating the label ZI unconstrained (ZI-U), who allow bids and offers to fall anywhere within the 200 value range. The second market restricted the ZI traders to a constraint causing them to be known as ZI with Constraint (ZI-C) where they were assigned a hypothetical value for stock, creating a volume tunnel, see section , of ‘costs’. In addition ‘if they generated a bid (to buy above their redemption value or an offer (to sell) below their cost, such actions were considered invalid and were ignored by the market’, therefore, bids and offers must all fall within the volume tunnel, creating profit or break-even transactions. The third market ran with the aforementioned human traders who having being told their individual cost values would be irrational to bid for potential loss making deals.

The difference between ZI-C and human bidders is now, it is claimed, based entirely on human rationality and the individual specific motivation to make the maximum amount of profit.

Five experiments with different clearing quantities and equilibrium values were performed and as can be seen by graph 2.9, which has been chosen as a representative of the five graphs which resulted from the experiments, there appeared some very distinct behaviour and intelligence. The ZI-U traders ‘exhibit little systematic pattern and no tendency towards any specific level’, behaviour is extremely volatile and as can be expected, no intelligence is displayed. The human traders show some initial, slight *tâtonnement*, section 2.2.3, but quickly converge to ‘settle in the proximity of the equilibrium price’ which shows clear learning and intelligence which is improved upon each day. The ZI-C traders are the most interesting however. They exhibit three clear learning behaviours which are denoted by Gode. Firstly that, unlike humans, they do not learn from trading periods because, as ZI traders, they cannot remember. Secondly, the degree to which the transaction appear erratic, though not as low as a human trader, is far below that of the ZI-U trader, which implies that ‘imposing a budget constraint on ZI traders is sufficient to shift the market performance towards the human market performance’. Lastly, the prices converge towards the end of each trading day. They have proved this mathematically by plotting ‘root mean squared deviation of prices from equilibrium’ on graph 2.10 from which they claim ‘by the end of a period, the price series in ZI-C trader markets converges to the equilibrium level almost as precisely as the price series from human trader markets does.’ As ZI agents which as specifically designed not to remember, adapt or learn, this convergence must be caused by ‘progressive narrowing of opportunity sets of ZI-C traders’. This is due to basic supply and demand principles, see section 2.2.1, as higher-value units are traded, more lower value units are available to be traded and *visa versa*. The offers at

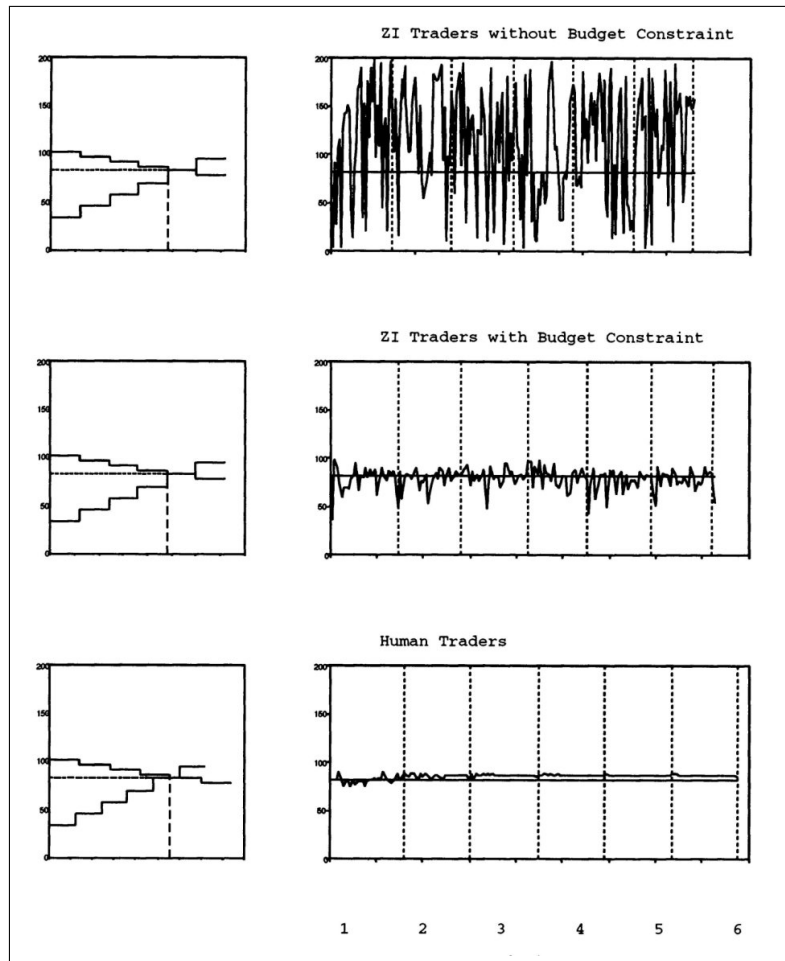


Figure 2.9: A typical graph displaying ZI-U, ZI-C and human results from an experiment. Supply & demand functions and transaction prime time series. Image courtesy of [7]

very high or very low prices are likely to make transactions first as they represent units on which the highest amount of profit can be made.

In conclusion, Gode claims that imposing restrictions on pricing is the only necessary procedure and that the convergence within results and the high allocative frequency were direct 'consequence of market discipline'. Even their critics had to admit the results were impressive as the average allocative efficiencies, see section 2.2.5, were 97.9% for humans and 98.7% for ZI-C's. The measure of profit dispersion, though the value was still greater, was much closer to the human results than the ZI results. This led to Cliff [5] stating that, with such small margins, the assumption that 'the high efficiency of the human market [as] a consequence of human cognitive prowess; in light of Gode and Saunder's results, such assumptions are clearly highly doubtful.'

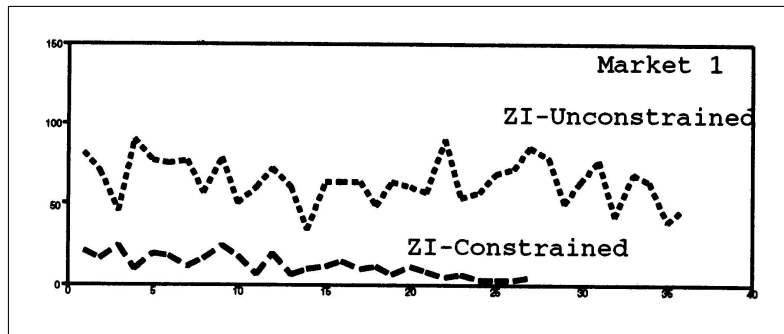


Figure 2.10: A typical graph displaying 'root mean squared deviation of prices from equilibrium'. Image courtesy of[7]

Yet Cliff[5] also points out several flaws in their arguments; 'maximum and minimum values for shout-prices have to specified in advance, which implies that [prior knowledge and] information is employed' which is against the privacy rules and that many of Smith's experiments possibly could not be preformed in this format.

2.5 Background Research Conclusion

Although many interesting issues have been raised by the papers reviewed, the most interesting one is contained in the abstract of the primary research article; Is convergence to equilibrium 'determined more by market structure than by intelligence of the traders of the market'[5]? In this report it is intended to examine human agents, ZI agents and implement some degree of intelligent market structure within the program. Some of the experimental scenerios reviewed in this section shall be recreated using the program and the results compared and contrasted in against this implementation in Chapter 6 to allow for critical analysis.

Smith experiments show human behaviour is unpredictable to an extent, although convergence was eventually reached, it was not always to exact equilibrium and often it took some time. When this is taken into account, how much intelligence is required of an agent to achieve human-level performance?[5] It is also worth noting the similarities between economic behaviour and animal/agent behaviour, although that will not be reviewed futher in this project, it could be considered to pose a question of how intelligent an animal is a human? When motivated for gain, it would appear extremely according to Smith [18]. Gode[7] critiques how Smiths [18]subjects were motivated to seek profit, yet when profit tâtonnement is implemented by both buyers and sellers this creates a equilibrium of forces[20] so that

the motivation to specially gain profit is irrelevant.

An interesting point is how Cliff concentrates on economics as a metaphor for computing in section[?], where most others [11], concentrate on computing to simulate, or be a metaphor for, economics. Interesting as research on economics is older, more complete perhaps?

Many papers reviewed [7],[18], suggest 'allocative efficiency very close to 100%', high degree of convergence and low profit dispersion without any prior intelligence or knowledge in simulated agents. If this can be reached during the program development, this will be a clear solution to the problem posed.

Chapter 3

Methodology

3.1 An Introduction to Methodology

A methodology is the technique of identifying activities to be preformed, planning methods to complete each individual task and arranging them to run in sequence or parallel in such a way as to produce an end product that works to specification while having utilised resources in an appropriate way. If this framework is followed, the project can be considered a success and hence the selection and application of a relevant methodology is an important part of the project development cycle.

There are many methodologies currently in use and each project uses a slight variation upon them depending on the requirements of the proposed system as not two projects are exactly the same. The conditions and properties which define this program as a project[8] are; planning is integral to success, there are strict deadlines, there is to be no 'end user', resources are fairly limited

This chapter will consider the key characteristics of the proposed project and review these against several potential established methodologies to decide on a tailored approach.

3.2 The Waterfall Model

The Waterfall Model is a generic guide which outlines the main stages of most projects in development. Originally presented as an example of a flawed non-working model [16], it can be applied in some scenarios with success as it outlines the five main steps of any project and requires a review at the end of each stage which means the process must be completed before progression to the next step.

3.1 forms the basis for many more complex structures, but as Royce himself claimed 'the [Water-

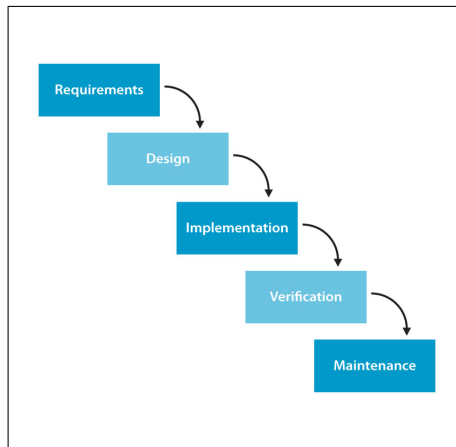


Figure 3.1: The Five Stages of the Waterfall Model, image courtesy of[9]

fall] Method has never worked on larger software developments’ because it does not allow for iterative behaviour. In any project, the scope to allow for improvements or modifications before completion of the project but not necessarily after each stage, for example, changes to the requirements may be necessary during the design phase, is important to success as is the case with the program in this report which makes this an inappropriate methodology.

3.3 The Spiral Model

‘An iterative systems development in which the stages of analysis, design, code and review repeat as new features’[2], this model is also known for reducing “risk” as when a stage of the project is under-developed, the next iteration is able to strengthen it until the problem is eliminated and hence the risk of failure is reduced’[14]. This is likely to be a useful model in the development of the project as it allows for changes in design, architecture and specifications as necessary, as well as absorption of features of other models[3].

3.4 Methodology Conclusion

It has been decided to formulate a custom model that incorporates the benefits of aforementioned development models, but allows the characteristics, as outlined in the ‘Introduction to Methodology’ such as the strict deadlines and limited resources, to take precedence. This means that although planning is indeed key, the 5 stages named in the waterfall model will be performed iteratively in much the way the

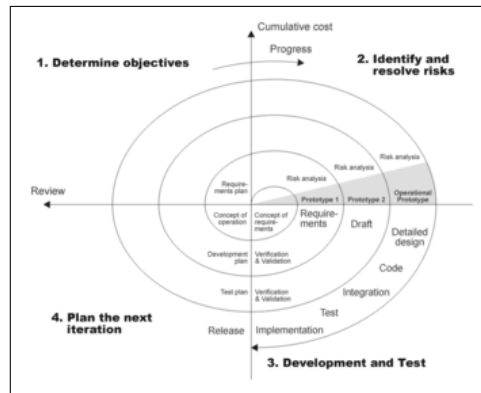


Figure 3.2: The Spiral Model, image courtesy of[1]

spiral model would, but allowing for flexibility should time run short. The use of both iterative and incremental changes during development should systematically give value to the final software deliverable.

As there is not enough time to do formal methodology in regard to write-up it shall be considered that a merged account should take place. This means that Design & Implementation shall not only be combined, but also appear in the chapters Testing Design, Results, which displays the design & implementation features and Evaluation which will consider how they could have been improved or why behaviours are such. Similarly, there should appear much of Evaluation in Results and visa versa as they are so integrated and dependant on each other. The use of SPE generated UML and screenshots shall be implemented, this is a method by which one can more easily visualise and document progress of software.

Chapter 4

Design & Implementation

4.1 An Introduction to Design & Implementation

In this section it will be explained how the program was designed and hence implemented. This will include justification of design decisions made, how the chosen methodology was used in obtaining the finished product and whether the minimum requirements were reached, these can be seen in section 1.4. In order to fully test the hypothesis it was necessary to exceed the requirements by creating an environment that implements learning bounds. This would be done by designing a trader known a Zero-Intelligence Bounded or ZIB.

Due the lack of resources, not least of which was the authors programming ability, explored further in section ‘Software Justification’ 4.3 it was considered appropriate that the program would be fairly small and the emphasis would be on producing results rather than coding style or dynamic programming. This allows for focus upon comparison and evaluation which is befitting for this type of project.

Having thoroughly researched possible approaches to this project, it was been decided to implement several features of the experiments and discard several others.....!!!!!!!!!!!!!!

Despite use of iterative methodology to design the program, it has been considered prudent to separate final testing from this section, but where relevant or design features have been made in response,reference will be made to testing preformed at each iteration.

4.2 Design Considerations

During the design phase, several features of the potential program had to be taken into account

- There was to be no prototyping due to the estimated size of the project
- There was to be no specific “end user”, only a facilitator to initiate the program
- The minimum criteria had to be implemented sequentially
- The minimum criteria had to be implemented in such a way as to remain compatible with previous research

In response, each of these was examined and solved in turn.

The lack of prototyping allowed for programming to be started immediately and instead versions were saved at various criteria or progress milestones. This was especially useful upon testing at the end of each iteration as it meant that if the wrong approach had been taken, it was simpler to retrocede to a previous working copy and progress from there.

In regards to having no end user, this is partially because it would be impractical to have a lot of outside influence which could affect results, partially to replicate previous experiments and partially to simulate an automated MBC, section 2.2.2 which is one of Cliff’s[5] aims. There is however a need for the program to be instigated and the conditions to be inputted, this makes a singular facilitator appropriate.

Having the minimum requirements implemented sequentially was not a problem to be solved, although it did mean that upon encountering issues, there was no alternative but to persist. However, this was beneficial for the program and hence for the project.

Keeping criteria compatible with previous research in order to simulate experiments was much harder than originally planned for and as is aforementioned, it was decided to implement several features of the experiments and discard several others in order to focus on the development of the ZIB trader.

4.3 Software Justification

Lack of prior programming knowledge and time in which to remedy this meant that selection of program was very important. A number of programs were examined with regards to how suited they were to both the project and the authors ability. The author has varying experience in the object-orientated programming languages Java, Visual Basic, PHP and MatLab, so these possibilities were explored first.

The design goals of Java[10], the most relevant of which, in this instance, is Simple, Object Orientated and Familiar make it a relatively easy program to learn and highly applicable for this program.

However there is was need in the project for flexible array sizes which Java could not provide and the strict syntactical structure made it impractical for the author.

Visual Basic provides excellent capabilities in regards to GUI and is the easiest available language to learn, but its high-level style made it too simple to fulfil requirements demanded in a final year project, even where software development is not the main objective.

PHP is built upon the PERL program so is good with string types, but this is not useful for this particular project. PHP is also geared towards internet use so is difficult to develop in and its use is highly discouraged by the University of Leeds with the result that this program was inappropriate.

Matlab is a complex mathematical program with good graphical applications, but the mathematics required would not utilise the capabilities of this program and also as an interpretive language, poor programming can cause it to run slowly. It was decided that this program was impractical to use in this project.

Having conducted research into this area it was decided to write the program either in C or Python.

The original Cliff program is written in C and had preformed extremely well. Its mathematical capabilities make it ideal for such an application. However it is a low-level language and as such very difficult to learn. Also the complexity of creating GUIs, which is of vital importance to this project, would create a significant amount of work and the time frame that was available would be unable to support the amount of learning involved. This, coupled with the authors lack of training made C an unsuitable choice.

Python is a very flexible, simple-to-learn, higher level programming language. It is capable of preforming mathematics to the degree required for the program and the tkinker import available allows for production of a reasonable GUI. Another important feature is that in addition to tkinker, there are several other support libraries which may be accessed to assist with standard module coding.

In conclusion, the features of python that make it good for quick product development make it ideal for this project [13].

Python has many integrated development environments, IDEs, available the most popular of which is the inbuilt IDE renamed IDLE. There are many useful features of this language such as syntax highlighting and an integrated debugger [15]. It has been suggest that 'Tkinter GUI and threaded programs may not work well with IDLE'[13] because IDLE itself is constructed in Tkinter.

It was decided to use SPE for several reasons. Firstly, the layout of it was similar to DrJava which the author was familiar and comfortable with. Secondly it has many of the features of IDLE including

autocompletion and PyChecker. Also useful was the automatic production of UML which has been displayed within this project and the source index which made code simple to locate and hence edit. In conclusion, to create a small program with only a reasonable amount of mathematical and visual requirements, it was the best choice.

4.4 System Architecture

As an Object-Oriented, OO, system, it was possible to take a modulated approach. This could be broken down into iterations easily which was extremely useful in terms of implementation as it meant each criteria could be designed, programmed and tested as independent modules. The original design was to create a class called Trader and use inheritance to pass information down to “children” classes, Buyer and Seller. This class structure can be seen in a UML class diagram in section 4.6. This basic architecture remained constant throughout.

As the transaction dictionary was implemented, a marketplace class was added and for the GUI, it was necessary to make the addition of an application class.

Using OO makes reference to concepts of traders as individuals as suggested in research done by Smith[18].

4.5 Mathematical Design

Experimental conditions were, where possible, mathematically be quite similar to the Cliff[5], Smith[18] and Gode[7] papers as possible in order to accurately recreate their conditions and hence have a fair experiment. The first condition was to set the trader format as a double auction in which ‘each bid, ask, and transaction was valid for a single unit. A transaction cancelled[sic] any unaccepted bids and offers. Finally, when a bid and ask crossed, the transaction price was equal to the earlier of the two’[7] It is worth noting that throughout this paper and in the ones researched, no ‘costs’ are incurred from bidding or during transactions.

4.5.1 Transaction Algorithm Design

In designing the transaction algorithm, there were several options to consider:

- Which trader type should be prioritised in making transactions?

- How should the two trader types be matched?
- At what price should the transaction take place?

A trader type must be prioritised because due to the design of the program, bids could not be made asynchronously. This meant time had no value as such because all bids and offers were made instantaneously and simultaneously. Each trader was within an array and as such it can be considered that the first value within the array would be considered the first trader to have made a bid or offer. Having made this assumption, it is fairly irrelevant whether the buyers or seller go first.

It is also an option to alternate priority, but this would have no mathematical or allocative advantages and would create excess coding for no gain. Therefore as was discussed in the introduction to this chapter, it was decided not to implement this technique.

There is a wide selection of methods available to match buyers and sellers.

One option would be to find the nearest match, but that may result in the buyer and/or seller entering into a loss making deal which has been seen in the experiments done by Smith [18] is unacceptable. As this necessity to enter into profit making deals has been recreated by Cliff [5] and Gode [7], so it shall be within this paper. Similarly, this necessity rules out the possibility of random matching.

If bids and offers had been assigned a time it would have been apt to enter into the first profit-making transaction which is encountered, for example a buyer to take the first seller with a lower offer. This would be an accurate representation of a real trader market and allow for simulation of experiments discussed in ‘Background Reading’ section 2.4. It also would allow for an element of machine learning as is the case in Cliff[5] where the subsequent bids and offers are based on the last transaction. This is not the case however and as such, the matching algorithm should attempt to be fair to emanulate the previous results as much as possible.

Transaction price also posed an issue as results can appear very different with even a small margin of difference in this field. Gode[7] posed his experiments by accepting the first price and this shall be simulated with an edited version of the code, however as the bids are not asynchronous, it was not appropriate for this program as it meant the list of transaction prices would always remain the same as the trader array which was chosen to be the base matcher.

It was decided to prioritise the buyers and compare each seller. If the seller price was lower it would be added to a shortlist, the maximum of this shortlist would then be chosen to make a transaction. This sets the transaction matching pattern to finding the closest profit-making match. The transaction price

would then become the mid-point of these two prices. This can be seen in line 10 of the code displayed in section 4.8. This was chosen for several reasons. The first being that, overall, it will not matter which trader type is given precedence, hence Buyer was chosen. Also as randomly chosen numbers the distribution should appear uniform and hence this matching pattern should maximise bids. If any remain, it should be that of the lowest buyer price, as there are less likely to be sellers the lower the buy price is, similarly with remaining sellers, their sell price should be the highest.

It could be argued that this is unlike a market in reality where a buyer would attempt to maximise their profit by finding the minimum value available and visa versa. However it could also be argued that a buyer offering for example to buy at four is unlikely to allow a different buyer offering six to make a transaction with a seller offering at two especially if there are no other sellers offering at a price lower than the first buyer if the bids are made simultaneously.

4.5.2 Bounded Transaction Algorithm Design

Creation of the Bounded Transaction algorithm was important as although it is beyond minimum requirements, it is vital to discovering if, by using this technique, convergence could improved.

The bounds to be implemented in this version of ZI-C/ZIP traders create a narrowing volume tunnel through stigmergic behaviour 2.3.1 - where past bids have been unsuccessful, the cost level at which this occurred will be removed. This is done by reviewing the Buyer and Seller arrays after a 'day' has occurred. Within these arrays, as noted in section 4.8, only the traders who are unsuccessful in their transactions should appear. The values of these traders should be as hypothesised in section 4.5.1 the lowest simulated values as buyers and highest values for sellers. These shall be sorted and the minimum and maximum of these values respectively found. This "minimum of the minimum" bids by buyers forms the lower bound of the next day whereas the maximum of the higher unaccepted offer will form the upper bound.

In implementing a random number generator to give standard distribution of values, the following code was used.

```
1 self.set_price(round(random.random() * (highbound - lowbound), 2) + lowbound)
```

This ensures that the lowest number is always on or above the lower bound and that the maximum value never exceeds the 'highbound'

4.6 Iteration 1

For the first criteria ‘Creation of ‘Buyer’ and ‘Seller’ arrays’, a base class known as ‘Trader’ was implemented with the subclasses ‘Buyer’ and ‘Seller’. Within each of these subclasses an empty array, known as a container was added. Diagram 4.1 shows this top-down architecture. The methods `set_price` and `get_price` can be seen within Trader class. These were originally coded as in Buyer and Seller subclasses, but it was decided to use inheritance to reduce replication of code and increase processing speed. The containers were then filled by appending to a loop that inserted pseudo random ‘`random.random`’ values and multiplied by 10 to allow them to fall between 0.00 and 10.00. The final step was to round to three significant figures to simulate decimal currency. It was done in this order because at the lowest level, a computer finds it easier to multiply, hence processor power is conserved. .

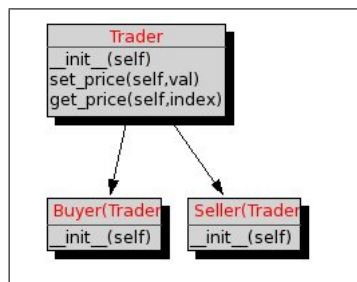


Figure 4.1: UML Model for Iteration One of Design & Implementation

4.7 Iteration 2

In order to create the second criteria, a ‘Transaction Dictionary’, it was necessary to consider what should be contained within. From graph 2.5 it can be seen that all bids, all offers and all successful transaction prices are saved plus the time at which they occurred. These are all details which are integral to realising patterns and behaviours and hence shall be included in the dictionary. As the program is computer simulated and the transactions take place in a matter of milliseconds, it is more appropriate to assign a sequential transaction number to each from which time can be inferred. Thus, having substituted values to be inputted with X, the dictionary contains:

```
1 transactiondictionary = {'transaction_number': X, 'transaction_value': X, '
    buyer_price': X, 'seller_price': X}
```

A loop was implemented in order to fill this with results from the transaction algorithm as described in sections 4.5 and 4.8.

4.8 Iteration 3

The third criteria was to design an algorithm that would match buyer and seller traders to perform a transaction. This was the most complex part of the program and required much adjustment before the next stage could be attempted.

The algorithm was heavily interlinked with the mathematical design, section 4.5, and as such required much consideration before it could be started and was designed and implemented much as section 4.5.1 states. Buyers were examined in turn and sequentially, sellers were then compared and the “maximin” or the maximum of the seller prices that were lower than that buy price chosen. If there were none available, the buyer was placed back into the array.

The values in the transaction array often appeared to 17 significant figures. This is highly inconvenient as it looks untidy and unprofessional, however after much experimentation and research it was discovered that this occurred because of computer representation of binary numbers and the only way to remove the excess digits would be to set the transaction dictionary as a string. As this would have been time-consuming and not entirely necessary, it was decided to focus on further programming instead. Line 10 of the code shows a partially successful attempt to correct this using ‘round’.

```
1      for buyer in self.buyerlist:
2          #find a seller for the given buyer
3          seller = self.match_seller(buyer.get_price())
4          if seller:
5              # if seller is found ,need to make a transaction for this
6              transaction = {}
7              transaction['seller_price'] = seller.get_price()
8              transaction['buyer_price'] = buyer.get_price()
9              #is the mid point of the above two values
10             transaction['transaction_value'] = round(seller.get_price() + ((
11                 buyer.get_price() - seller.get_price()) * 0.5),3)
12             transaction['transaction_number'] = len(self.transactions) + 1
13
14             self.transactions.append(transaction)
```

```

15         #remove the buyer and seller from the retrospective lists
16         self.sellerlist.remove(seller)
17         buyer_remove_list.append(buyer)
18
19     #remove buyers that completed transactions.
20     for buyer in buyer_remove_list:
21         self.buyerlist.remove(buyer)

```

As can be seen from the code lines 15-17, the buyer and seller involved in the transaction are removed from the list to ensure that they are no longer capable of making further transactions. This is because each trader is only assigned a singular unit of stock and in keeping with the double auction format as described in previous research [7] outlined in section 4.5. A fortuitous benefit of this is that upon removing each trader involved in transactions, the remaining traders could be displayed using a simple 'print array' command.

The methods used in implementing this dictionary and algorithm were kept inside a 'Market' class. This was effectively the virtual version of an order book and can be seen as an independent class in figure 4.2. As can also be seen, it has been amended so that arguments and keyword arguments, 'def __init__(self, *args, **kwargs)' were used in the definitions of buyer and seller to allow for variable number of attributes. This gives increasing flexibility of call signatures which is useful when the GUI is implemented in Iteration 4 and the bands are implemented in Iteration 5.

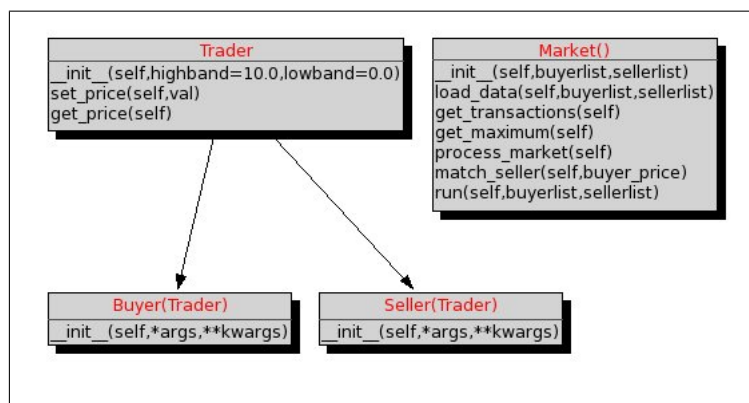


Figure 4.2: UML Model for Iteration Three of Design & Implementation

4.9 Iteration 4

Criteria four, to create a GUI which displays past transactions, was also extremely difficult to implement.

The importable Tkinter library was used in conjunction with a new class named Application whereupon the first task was to create the visual aspects of the GUI. This was done by using a Frame Widget within which a canvas with the specification:

```
1 self.canvas = Canvas(width=600, height=512, bg='white')
```

and the buttons 'Quit', 'Run' and 'Run with Bounds' appear. Ten lines were drawn on each axis of the canvas, individually labelled with values originally set to 0.0. As is traditional in economic graphs price is on the vertical axis against the horizontal time axis. A static point was "drawn" in the (0,0) position so that no lines would appear on the graph until the 'Run' button was hit, initiating the sequence. This results in figure 4.3.

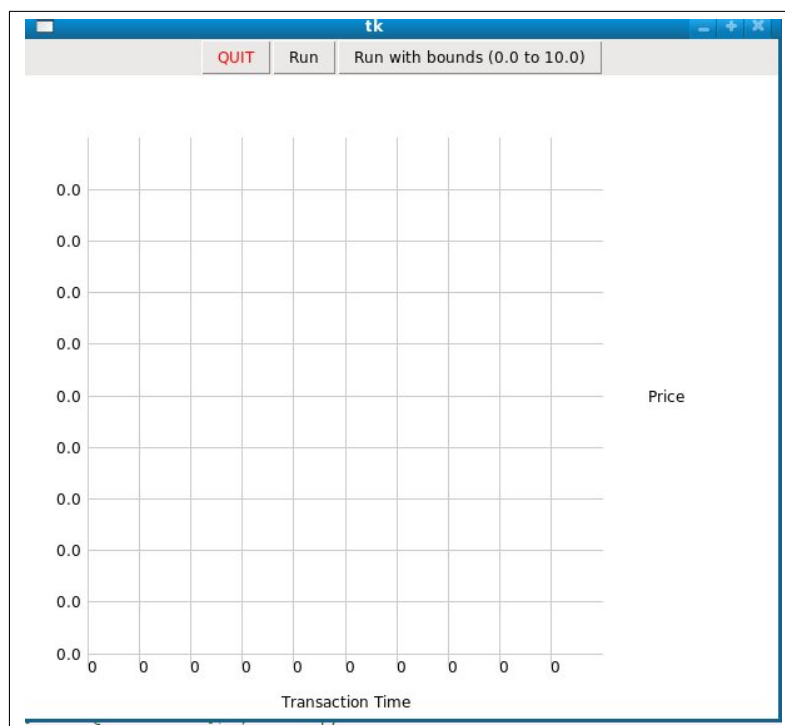


Figure 4.3: Screenshot showing an empty Tkinker GUI

Application then uses an 'if' loop around `market.get_transaction`. Upon receiving the transaction, a point is drawn upon the graph, its 'y' axis height dependant on price value. The value of the 'x' co-ordinate is based upon order and standardly distributed as its relation to time would suggest. This

is a similarity with the graphs shown by Cliff [5] and allows for compression when the application is required to show a few ‘days’ of trading, which is necessary for this project. This technique is also used by the ‘y’ axis upon which the values expand to the highest previous value displayed. The ‘if’ loop is now completed to initiate a create_line function between the previous point and the current. This is iterated until either the range is reached or the flow of transactions halts, upon which it is required to press one of the aforementioned buttons in order to instigate a designated process. Upon pressing ‘Run’ several times, a graph similar to figure 4.4 will be shown.

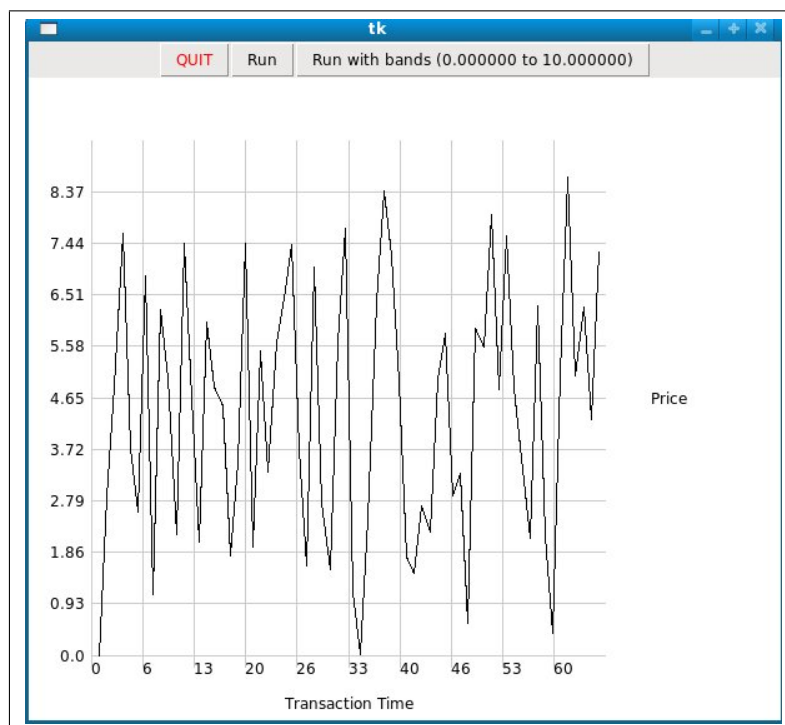


Figure 4.4: Screenshot showing a Tkinter GUI with data fill

4.10 Iteration 5

As with Iteration 3, the design of this section was heavily dependant on the design of the bounded transaction algorithm and hence the design is as described in section 4.5.2. It was chosen to place the algorithm within the Application class with the reasoning that it would be more beneficial to view convergence to equilibrium upon a graph than via numerical data.

The method run_with_bounds was implemented and others amended, this changed the UML to appear as 4.5.

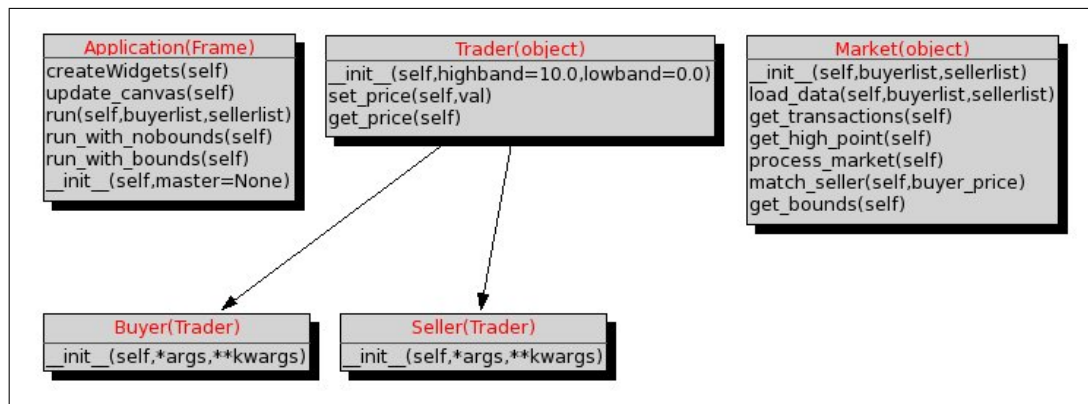


Figure 4.5: Screenshot showing the UML of Iteration 5

This iteration was not completed in the sense that, although bounds were implemented, there was not enough testing and refining done and as such, the technique is not as effective as it should have been.

Chapter 5

Testing Design

5.1 Introduction

To be able to evaluate this program effectively will require thorough testing.

Reviewed previous research has also suggested several useful and interesting tests that could be preformed to allow the ability to compare and contrast. Also to enter previously untested boundaries in order to review varying effects on pricing.

5.2 Minimum Criteria Testing

5.2.1 Test One

The first of the minimum criteria shall be tested; can the program create buyer and seller arrays containing random values.

This test shall be carried out with a number of values. First with array size six, then ten, then one hundred. This should be adequate to establish whether the basic program features are capable of the tests required. If this test is not successful, the remainder of the project cannot be tested.

5.2.2 Tests Two & Three

The second criteria was the creation of a 'Transaction Dictionary'. This should store the buyer bid, seller bid, transaction value and a transaction number.

The third criteria, the creation of a transaction matching algorithm, is also an intergral part of the

project. This allows the transactions contained within the dictionary to occur hence, these two tests will be carried out simultaneously. These, too, shall be carried out with a number of values as, again if this basic design feature does not perform then the project cannot continue in the method planned.

5.2.3 Test Four

The fourth and final minimum criteria was a GUI capable of displaying past transactions. This is not vital to the running of the program, but is important in terms of being able to effectively evaluate the progress of the bidding and observe the level of intelligence. This test shall be performed with ZI agents to primarily test its ability but the results displayed should also allow for evaluation in Chapter 7.1.

5.3 Extension Tests

In order to accurately analysis the theory that implementing bounds, based on past unaccepted transaction bids and offers, creates a learning environment that can simulate human behaviour, the further extension of bounding should be extensively tested. As is stated in design section 4.10 however, this is an unfinished iteration so it would be sensible to perform three tests of five days, in the style of Cliff[5] and consider these to be typical. One will be shown within the project, the remaining two in Appendix B.1.

5.4 Human Trader Tests

This is something that can be recreated easily and provides a distinct contrast to test four which will display zero-intelligence. The cognitive powers of human trading will give a benchmark against which the ZI traders and ZIB traders can be compared. Viewing these patterns in action delivers benefits for the project in the form of feedback from the test subjects and the ability to customise the experiments to exactly match the conditions within which the program will work.

5.4.1 Smith Tests

Smith performed tests with human traders to prove his theories. These shall be done by first recreating the conditions and boundaries to which Smith worked, then carried out within the Human Trader Experiments.

Chapter 6

Results

This section has two functions; to display physical data that is outputted by the system and then to compare and critique it against expectations and similar data found previous research.

6.1 Minimum Criteria Results

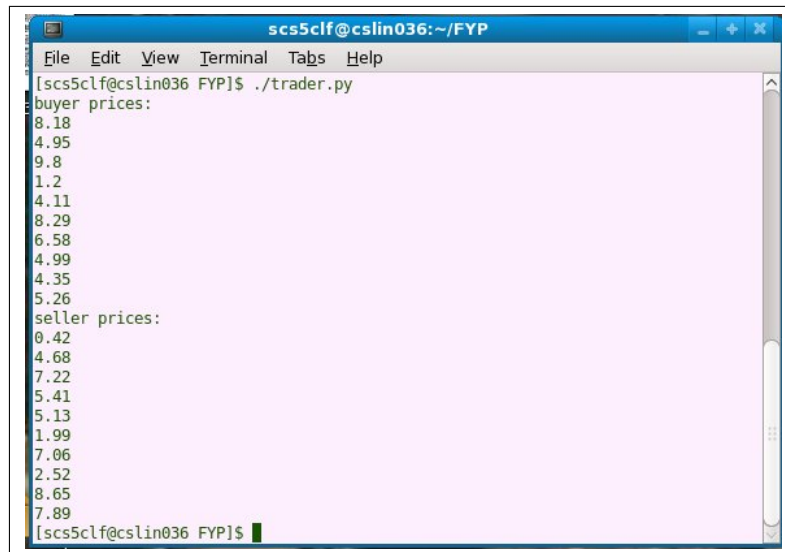
This section displays the results from the `trader.py` program as described in section 5.2. Screenshots shall be used to display results and where multiple tests have been preformed, other tested sets of values appear in Appendix B.

6.1.1 Test One

The first of the minimum criteria that was tested, can the program create buyer and seller arrays containing random values, was preformed successfully.

As was outlined in section 5.2.1, it has been tested with several sets of values. A screenshot containing results from running the program with ten values is shown here. Screenshots showing other tested sets of values appear in Appendix B, section B.2.1. As can be seen, the program is capable of handling varying sizes of array hence it can be considered that criteria one has been fulfilled.

The numbers appear reasonably distributed and random so expectations have been met, however this has not been accounted in precious research so it is impossible to compare.



```
scs5clf@cslin036:~/FYP
File Edit View Terminal Tabs Help
[scs5clf@cslin036 FYP]$ ./trader.py
buyer prices:
8.18
4.95
9.8
1.2
4.11
8.29
6.58
4.99
4.35
5.26
seller prices:
0.42
4.68
7.22
5.41
5.13
1.99
7.06
2.52
8.65
7.89
[scs5clf@cslin036 FYP]$
```

Figure 6.1: Screenshot showing ten randomly assigned values to each of the buyer and seller arrays. Results from Test One

6.1.2 Tests Two & Three

As can be seen in Figure 6.2, first the buyer and seller arrays are displayed. This is important in order to clearly see what prices are available for transactions and hence be able to evaluate performance better.

From these buyer and seller arrays, transactions have been matched as planned in Design & Implementation, section 4.5.1. As was predicted the remaining buyer price of 0.16 is the lowest and the seller price of 9.23, the highest. These would make appropriate bounds with which to narrow the feasible region of randomly simulated numbers.

The two criteria that are tested here can be considered to be successfully completed to a good standard. There are some issues which could be improved, but these shall be explored further in section 7.2.1.

As was stated in design, the reason the attributes within the dictionary were chosen is partially to allow them to appear compatible to the results of other data, thus expectations have been met.

6.1.3 Test Four

The fourth and final minimum criteria was a GUI capable of displaying past transactions. As can be seen from figure 6.3, this is not only possible, but the program is fairly flexible. Consider figure 4.3 with its empty data set, then compare with figures B.3 and 6.3. The method of compression is useful and effective and allows for simulation of previous research.

```

[scs5clf@cslin039 FYP]$ ./trader5.6.py
buyer prices:
4.89
0.16
1.74
8.07
8.6
9.05
9.01
4.04
3.49
7.77
seller prices:
0.58
0.68
0.61
6.0
4.42
6.95
2.52
2.46
4.37
9.23
[{'transaction_number': 1, 'transaction_value': 4.629999999999999, 'buyer_price': 4.8899999999999997, 'seller_price': 4.3700000000000001}, {'transaction_number': 2, 'transaction_value': 1.175, 'buyer_price': 1.74, 'seller_price': 0.6099999999999999}, {'transaction_number': 3, 'transaction_value': 5.264999999999997, 'buyer_price': 8.0700000000000003, 'seller_price': 2.46}, {'transaction_number': 4, 'transaction_value': 5.559999999999996, 'buyer_price': 8.599999999999996, 'seller_price': 2.52}, {'transaction_number': 5, 'transaction_value': 8.0, 'buyer_price': 9.0500000000000007, 'seller_price': 6.9500000000000002}, {'transaction_number': 6, 'transaction_value': 6.714999999999999, 'buyer_price': 9.009999999999998, 'seller_price': 4.419999999999999}, {'transaction_number': 7, 'transaction_value': 2.359999999999999, 'buyer_price': 4.04, 'seller_price': 0.6800000000000005}, {'transaction_number': 8, 'transaction_value': 2.0350000000000001, 'buyer_price': 3.4900000000000002, 'seller_price': 0.5799999999999996}, {'transaction_number': 9, 'transaction_value': 6.884999999999998, 'buyer_price': 7.769999999999996, 'seller_price': 6.0}]
unmatched buyer prices:
0.16
unmatched seller prices:
9.23
Transactions taken place 9
[scs5clf@cslin039 FYP]$ █

```

Figure 6.2: Screenshot showing ten randomly assigned values to each of the buyer and seller arrays which result in a transaction dictionary containing 9 transactions with the remaining two traders clearly indicated. Results from Tests Two & Three

The results shown on figures 6.3, B.4 and B.5 are interesting in regards to the theory posed in the introduction of this report. Some very slight convergence is noted at the end of each trading day, with the majority of transactions occurring between 3 and 7. This is reminiscent of results found in the ZI Traders without Budget Constraint graph that is shown within Godes [7] research on figure 2.9 where in response he claims his ZI-U traders 'exhibit little systematic pattern and no tendency towards any specific level'. The slight contrast between the ZI-U researched by Gode [7] and the ZI traders explored here are that the transaction algorithm is designed to display less volatile behaviour by taking the mid-point of the two price rather than the first one 'shouted'.

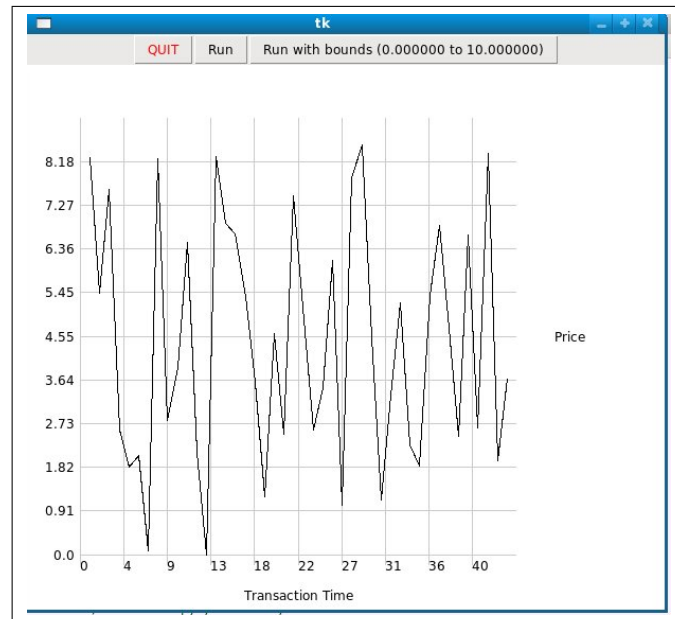


Figure 6.3: Screenshot showing a Tkinter GUI with 5 days of trading with ZI agents. Results from Test Four

6.2 Test Five

This section shows results from the testing of the bounding algorithm. Results are effectively displayed with the bounds printed upon the ‘Run with bounds’ button. This is some small success.

As is previously stated, this technique has not been perfected and this is reflected in the results. As can be seen in figure 6.4 within the project, and figures B.6& B.7 in Appendix B.1, convergence is indeed reached, in the cases of figure 6.4& B.7 to almost exact equilibrium. In reaching this, however, the bounds often close up completely causing some runs not to be able to reach the five ‘day’ testing requirement as shown in all the figures referred to here. This error plus that the bounding values that are displayed upon the ‘Run with bounds X to X’ button often do not match up to the bounds that are displayed onscreen suggests mathematical errors in the code.

6.3 Human Trader Test Results

In order to fully realise the results of the program, it was necessary to recreate the experiments done as performed in Cliff, Smith and Gode by gathering ten human graduates to operate fair tests. In each the trading ‘day’ was 4 minutes and on the graphs the first character of the timeline references this day

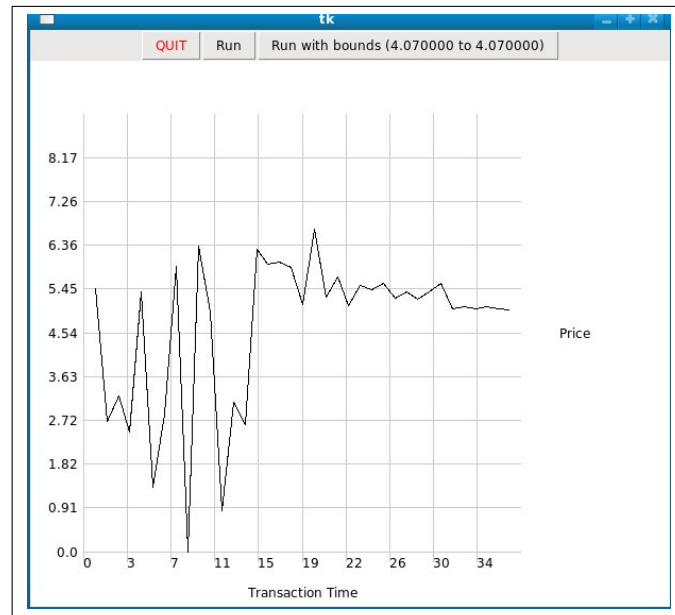


Figure 6.4: Screenshot showing a Tkinter GUI with 5 days of trading with ZI agents. Results from Test Five

number. As a first, control test, having been split into two even groups of buyers and sellers, they were assigned redemption values of:

- Sellers: A= \$1.50, B=\$2.00, C = \$2.50, D = \$3.00, E = \$3.50
- Buyers: F= \$1.50, G=\$2.00, H = \$2.50, I = \$3.00, J = \$3.50

As seen in the supply and demand section,section 2.2.1, this gives a clearing quantity of 3 and an equilibrium value of 2.5 , it was stated that it was acceptable to enter into loss making deals, but that this would be added to the cumulative frequency of profit at the final scoring. The results have been rounded to 2 significant figures.

The graph 6.5 shows the results of this test. As was expected,

It shall also be tested against the bounds of the program itself. This involves even groups, without redemption values and setting wide bounds of 0-10. Using these wide bounds disregard entirely 'market structure' as advocated by Gode [7] and display pure intelligence.

Graph 6.6 shows some intial erratic behaviour as some traders claim they wish to “test the water” with bids that would reap the most profit, only to find that these were not accepted as ,even at the intial stage of trading, no-one was willing to accept marginal profit. The transaction behaviour is much more consistent as there are two traders involved in deciding price.

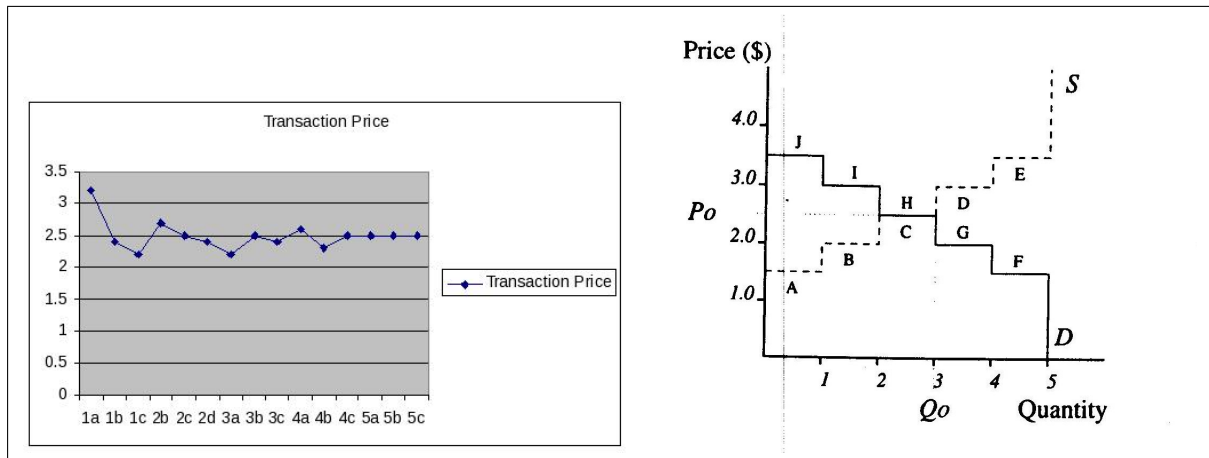


Figure 6.5: A graph displaying 5 days of human trading activity with bounds. Second graph shows theoretical equilibrium value of traders, see graph 2.2

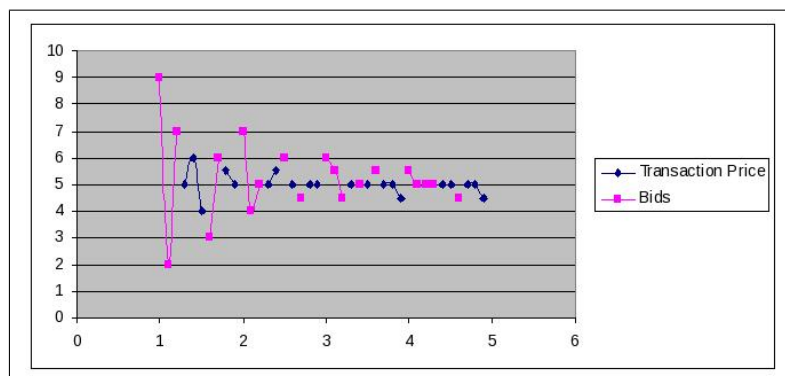


Figure 6.6: A graph displaying 4 days of human trading activity simulating the project conditions without bounds

In conclusion to trends of human trading, as can be seen, as with previous results explored from Smith and Gode, no-one was willing to enter into loss-making deals, the clearing quantity was reached each day and there is clear, speedy convergence to equilibrium. Also there is obvious day-to-day learning, some of which can be attributed to the subconscious human implemented bounds. This was confirmed when several participants questioned stated that they were unwilling to make bids far from the equilibrium, knowing that these were unlikely to result in a transaction. Although individual transactions can appear erratic, ‘should not impute all observed irrationalities of individuals to markets or impute all rationality of markets to their participants’ [7] These are standard qualities that appear in all human trader graphs. It is interesting to note also that the patterns within the graphs suggest swarm

intelligence and emergence, section 2.3.2 in the 'emerging' behaviour shown. A slight trend to occur, that has previously not been noted in research contained within this paper, is the last minute bid. This appeared a couple of times where a trader has waited until the last possible moment to exchange and refused to negotiate. The trader whom is being negotiated with then has to accept the bid at a non-equilibrium price and take a small dip in profit or to receive no profit at all. This is a method of game theory[6]. Individual transactions can appear erratic, 'should not impute all observed irrationalities of individuals to markets or impute all rationality of markets to their participants' [7]

Although individual transactions can appear erratic, 'should not impute all observed irrationalities of individuals to markets or impute all rationality of markets to their participants' [7] These are standard qualities that appear in all human trader graphs.

Chapter 7

Evaluation

7.1 An Introduction to Evaluation

Within this chapter, the finished program shall be evaluated in several ways. It is intended to evaluate the program as whole, including whether design decisions were correct, whether methods implemented where necessary and how the project could be improved. It shall first be compared to the minimum criteria as these have been justified in that they were the original plan for this project and have been proved to be a worthy guide to inspiring schedules, research and ,most importantly, ideas. The further development shall be explored as this has also been an important feature of this project and also quite interesting.

The relevance of project report shall be examined as will the degree of success the program has in solving the problem scenario. Methodology will be looked at in regard to how the program and report were created and the evaluation will finish with a brief summary of how the experiments within this report have shown patterns of interest and perhaps of importance.

7.2 Overall Evaluation of Program

As has been stated previously, the design of this program prohibited the recreation of the exact conditions of the experiments explored in ‘Background Reading’, section ??, but allowed for the conduction of experiments within the same scenarios. Although the plan was to re-create the experiments, many comparisons have been able to be made from the results contained within the papers regardless. This plus the fact that minimum requirements were met and exceeded means that this project is, overall,

successful. The best feature of this project is the use of ZIB agents which is quite possibly an original concept in the way it has been implemented and although it requires further development, initial results look promising.

If the bidding design by Cliff[5] had been recreated, this project would have been very different. Whether it would be an improvement will remain ambiguous, but it would have been simpler to apply patterns and behaviours and the relevance of the report could have been improved.

7.2.1 Evaluating program against minimum requirements

The minimum requirement were all successfully reached as can be seen in section 5.2

Tests two and three show that those criteria two and three were particularly successful in their format, however, it was seen that rarely was 100% allocative efficiency reached. This is inconsistent with the ‘very close to 100%’ efficiency reported by Cliff[5], Smith[18] and Gode[7]. This is, in part, due to the bid and offer process implemented. Whereas the aforementioned researchers had asynchronous bidding with transactions occurring during the ‘day’ as opposed to how this program creates the transactions at the end. Having transactions occurring during the day allows for an element of “haggling” and thus, encourages convergence. This could have been implemented if the main body of research had been completed before implementation took place as it would have been understood that this was what was necessary to effectively create simulations of past research.

Criteria four was also reached to a high standard as it does display past transactions well and looks clear. Although it could have been improved in relation to simulating past experiments by having a static ‘y’ axis showing the values 0-10.

Although all minimum criteria were fulfilled, it did not have the ability to recreate the experiments discussed in chapter 2.4 in a way as to allow for a fair experiment. This is primarily due to the way the arrays were filled with values in criteria one. Although not as issue in itself, the lack of asynchronous bidding meant that the remainder of the criteria could not create the bidding conditions as in previous research, but it was, however, possible to recreate scenarios which was done successfully in several tests 6, particularly ones conducted with human traders.

7.2.2 Evaluating program against Further Development

Despite it being necessary to implement the ZIB trader in order to test fully test hypothesis, it was possible to test it mathematically or with human traders. As a result, it was decided not to add this to

the list of minimum criteria that were decided in the mid-term report and instead shall be considered a further development.

Despite much research and experimentation, the bounding method could not be perfected. This may be partially due to implementing the methods inside the Application class whereupon it was more complex to access values. This refers to both the difficulty the algorithm had in gathering array and dictionary values and for the user to view how the transactions and bounds were occurring.

The bounding algorithm can be viewed as partially successful as, to the authors knowledge, the concept of the algorithm was original and sound. When viewed theoretically there is no reason why this method should not work in causing a ‘progressive narrowing of opportunity sets’ as with Gode [7], but without the prior knowledge that is critiqued by Cliff[5]. This is as it is the market that is being narrowed rather than the buy/sell prices of individual agents. This can be regarded as adding to the field as it is a combination of the two factors suggested by the papers mentioned priorly; it creates a market structure that implements learning and intelligence.

7.3 Evaluating Possible Improvements

Although the program works to a good standard and it is possible to evaluate from the results produced and the further work allowed for production of hypothesis, the program could have been improved by being more user friendly and workable, i.e.. results all in one place or trading ‘days’ clearly marked upon the GUI.

Other improvements could be that mathematical design could have been amended to raise allocative efficiency. This could also have been improved by allowing traders without transactions to continue bidding. This does become complex as to where to stop bidding, or to implement the end of the ‘day’.

Bounding technique could have been far improved by implementing Japanese Stock Market rules whereby markets may only change by a certain amount of “points” a day. Also by only implementing the upper bound for seller prices and the lower for buyers. This would have more accurately recreated the volume tunnels as implemented in the previous research.

7.4 Evaluating Relevance of Project Report

Having reviewed the background reading, none of the information was superfluous as many of the patterns and traits discussed appeared in the results or were relevant in deciding the design of the pro-

gram. Several behaviours written about were quite surprising in the way they appeared for example in the ‘Background Reading Conclusion’ it was noted that Cliff [5] used economics as a metaphor for computer simulations. Throughout this project, it has been interesting to observe how applicable the economic patterns are onto the virtual agents despite the little control or outside rules governing behaviour as hence proves Cliff’s point.

In regards to real-life, the problem was one that researchers are attempting to solve every day and programs are constantly being updated to try to predict behaviours through simulation. This

This report is fairly conclusive and self-contained. It would be considered to be comprehensive, but as the exact simulations done by Cliff [5] and Gode [7] have not been re-created, it cannot. However the results the researchers themselves received have been fully explored and comparisons and contrasts shown against these original results so it could be considered to be fairly comprehensive.

7.5 Evaluating use of Methodology

There was not enough time to follow any precise methodology, but the customised one gave an appropriate structure which was especially useful during the implementation and writing this report. The use of milestones to evaluate against in design whereupon difficulties could be discussed at each stage left the project well structured.

The methodology that was used in selecting background reading was followed somewhat unsuccessfully. As a fairly self-contained paper, Cliff [5] successfully covered many topics vital to the understanding of the project, but the two papers referenced within, when read in their primary format, created the need for unnecessary and time-consuming research.

7.6 Evaluating success of Solving the Problem Scenario

Due to the constantly changing, dynamic world economy, it is unlikely that programs such as the one created for this project would have much real-world success in an atmosphere such as a stock market or even the retail industry. As Smith succinctly states ‘The experimental conditions of supply and demand are held constant over several successive trading periods in order to give any equilibrating mechanisms an opportunity to establish an equilibrium over time. Real markets are likely to be continually subjected to changing conditions of supply and demand.’ [11],[18] This considered, this program cannot be viewed as a solution to the real-world problem of predicting trader patterns.

However, within the static market that was created, convergence can be seen to be reached, in some cases to exact equilibrium, with the bounded method as displayed in section 6.2. As the theory behind the concept was sound and discussed in section 4.5.2, there is no reason why, with further development, convergence could not be reached in a fairly fast and effective way.

7.7 Evaluating Human vs ZI vs ZIB

The three main experimental agent types conducted within this project have been human, ZI and ZIB. These shall be briefly summarised.

Traits of human traders, as has been seen in the experiments conducted in section 6.3 can be temperamental, yet display fascinating pattern and behaviours.

It is possible for humans to act irrationally in terms of reaching equilibrium, whether this is due to feeling “cheated” in previous rounds and consequently refusing to co-operate as seen in the former, or whether someone is cheating to maximise their own profit in the latter. Yet they also show much of the agent behaviours described in section 2.3 and the speed at which human cognitive power works in recognising equilibrium and recreating economic patterns is particularly impressive. Recreating the basics of the Smith experiment was probably the most beneficial of all the tests available, due to the ability to view the bidding procedure first-hand and receive qualitative feedback which could be explored without the need for speculation on the reasons for behaviour.

Traits of ZI traders have been uniform in behaviour or lack of it, except where simple rules have been implemented. Without rules, Smith [18], Gode [7] and this report have found them to be highly volatile as one might imagine given their utter lack of co-ordination and learning ability.

ZIB traders is a method which has the potential to be successful, initial results look promising due to their rapid convergence and lack of controversial conditions. Ultimately exploring the results and potential of this agent was interesting and it is this, more than anything else in this project, that is a success.

Bibliography

- [1] accuracyandaesthetics.com. Spiral Model, 2007.
<http://accuracyandaesthetics.com/tag/spiral> (Last accessed on 1st April 2009).
- [2] Greasley A. & Hickie S Bocij P., Chaffey D. *Business Information Systems*. Pearson Education Limited, 3rd edition, 2006.
- [3] Barry W. Boehm. A spirial model of software development and enhancement. 'Computer', 'IEEE', May, 21(5):61-72, 1988.
- [4] Eric. Bonabeau. Social insect colonies as complex adaptive systems. 'Ecosystems, Vol. 1, No. 5 (Sep. - Oct.), pp. 437-443, 1998.
- [5] Janet Bruten Dave Cliff. Minimal-intelligence agents for bargaining behaviours in market-based environments. Agency & Mediated Communications Department, August 1997.
- [6] Charles A. Davis, Douglas D. & Holt. *Experimental Economics*. Princeton University Press, 1993.
- [7] Sunder S. Gode, D.K. Allocative efficiency of markets with zero-intelligence traders:market as a partial subsitute for individual rationality. The Journal of Political Economy, Vol.101, No.1, pp. 119-137, Feb 1993.
- [8] Mike Hughes, Bob & Cotterell. *Software Project Management*. McGraw-Hill Education, 4th edition, 2006.
- [9] Hydro4ge. Waterfall Model, 2009.
Available at <http://blog.hydro4ge.com/?p=15> (Last accessed on 1st April 2009).
- [10] Java.sun.com. Design Goals, last accessed 26th March 2009.
<http://java.sun.com/docs/white/langenv/Intro.doc2.html>.

- [11] Alvin E. Kagel, John H. & Roth. *The Handbook of Experimental Economics*. Princeton University Press, 1995.
- [12] Glazer Amihai & Hirshleifer David Kirshleifer, Jack. *Price Theory and Applications*. Cambridge University Press, 7th edition, 2005.
- [13] Mark Lutz. *Learning Python*. O'Reilly Media, 3rd edition, 2008.
- [14] Steve McConnell. *Taming Wild Software Schedules*. Microsoft Press, 1st edition, 1996.
- [15] python.org. *IDLE Libuary*, last accessed 22th April 2009. <http://docs.python.org/library/idle.html>.
- [16] Winston. Royce. Managing the development of large software systems. Proceeding of the IEEE Journal, August, Vol. 8 p1-9, 1970.
- [17] Adam Smith. *The Wealth of Nations*. W. Strahan and T. Cadell, London, 1776.
- [18] L. Smith, Vernon. An experimental study of competitive market behaviour. The Journal of Political Economy, Vol.70, 111-137, 1962.
- [19] L. Smith, Vernon. *Papers in Experimental Economics*. Cambridge University Press, Cambridge, 1992.
- [20] Walras, Leon,translated by Jaffe, William. *Elements of Pure Economics*. George Allen and Unwin LTD, 1954.

Appendix A

Personal Reflection

Overall, I think my project went well.

The project selection was changed during the write-up. Originally the aim was to recreate the experiments done by Cliff, Smith & Gode, but as the project was written with the benefit of hindsight, I realised that I had created a ZI-agent that was capable of convergence without prior knowledge. This allowed me to write a hypothesis that would allow my bespoke agent to compete with the agents suggested within the papers written by Cliff[5] and Gode[7]. I feel that this made it a more interesting project, despite the likeliness of my agent being superior! Having tested the agent thoroughly, results were encouraging but it is highly unlikely that it could compete on quality to the aforementioned papers.

This lack of focus within my report is probably the detail with which I struggled with the most. Although later on, it was established that I would create a ZI trader rather than recreating the ZIP and ZI-C traders. As an evaluation project which concentrated mainly on research, the small change in project focus is not a disaster, but did mean many changes in approach had to be taken and due to time constraints this was perhaps not done to the extremely high standard which is required.

In regards to project topic selection, I enjoyed learning about Economic behaviours and models as this is something, I feel, I normally would not have had the opportunity to experience, and as a result have decided to continue my education in this field. However, with hindsight I realise that to adequately summarise and evaluate such a huge field requires a level of background knowledge that frankly I hadn't gained during my two years of taking Business as an interdisciplinary study. Papers that were helpfully recommended by Dr Dan Ladley (whose own PhD inspired this project) gave me a broad background of research possibilities, but as was previously stated, the lack of focus until the eleventh hour meant that time was wasted going into depth on subjects that need only be briefly explained.

Similarly, but to a lesser extent, was the level of AI that I had wished to implement. However, although I didn't create AI to the exact specification required to exactly simulate the past experiments, I felt that what has been created is appropriate to the scope of the project and as, ultimately, it works, I am pleased with my project selection overall.

Time apportionment was fairly appropriate as the original schedule calendar showed the agenda to begin at the beginning of Easter giving 43 days to complete the write-up. This should have been sufficient, but the time taken to complete the chapter 'Background Reading and Research' was disproportionate to the project. Although much of what was contained within was important to the understanding, design and evaluation of the project, it would have been far more efficient to conduct write-up during the research stage. This would have provided viable evidence of what was completed and what was not understood as during explanation, I realised that many of the concepts that were vital to fully comprehend the problem situation needed to be explored in the report and this was time consuming.

With hindsight I realise that I should have kept a blog or diary of the design of my program. Although copies were saved after each criteria milestone was reached, I had not commented on the problems I had incurred nor written where design had differed from the original specifications. Having left over a month between programming and Design write-up, this meant I had forgotten many of the intricacies for which the brief comments on which I had left did not explore to the depth that this report requires. Recalling these details and fixing the bugs whilst writing the design section took up valuable time close to the deadline and the rest of the project suffered.

Regarding allocation of work at supervisor meetings. I turned up to all, was generally prompt and made optimistic, yet achievable, schedules at each assisted by supervisor. These were not often followed precisely and I feel that if I had followed the timetable laid out by my supervisor and myself, I could have achieved a superior project and report.

The most important thing I learnt from the project report was the vital skill that is time management. Initially, although interested in the topic, I was easily distracted from the task at hand, this hindered my project in several ways. Firstly, I didn't understand the project properly until it was too late to actually enjoy learning and having finished the implementation before completing the research caused me to change focus as described above. Also it meant working up until deadline which means it is possible that silly mistakes have been made. I did however organise my modules into an 80-40 split knowing that this would allow me the whole of Easter to devote solely to this project. This was massively helpful and would be recommended for last minute workers.

Advice to students wishing to emanulate this project:

- Timing. There is a reason why this is on every report and that is because it really is the most important part of a project. A beautifully designed project will get few marks if barely anything has been written or explored, and conversely a poorer project will get reasonable marks if it is well explained. Make targets and deadlines and stick to them.
- Choose a topic with a focus and scope that is appropriate to your skill set. If programming is your forte, choose software engineering, similarly with if you are more comfortable with deep research, choose a subject that has potential to be theoretically developed and do an evaluation project.
- Prioritise. With a limited amount of resources available to you (ie. time, page limit, motivation), it is easy to lose focus and waste these resources on irrelevant diversions. Think about what you will need in order to succeed. Write in bullet points which can be built upon and provide regular intervals to ensure you have a fair representation of what percentage you have done and how much you will need to do.
- Do not be afraid of your project. It may seem intimidating at first, but once you understand what you are doing and things start moving, you may find you actually enjoy the ride!
- Once you understand something, write it down. This has several benefits. Firstly, it may make you realise you do not understand it to the depth you think that you do, secondly if you feel you understand something, you may be tempted to leave it for a while, having notes will remind you quickly and clearly without having to do all the research again. Lastly, each word written during the preliminary research or design stages is one word less to write the day before the deadline. Your fingers will thank you for it come April.
- Pick a topic that you enjoy. This will keep you motivated when the research, programming or report gets tricky.

Appendix B

Appendix B -Test results

B.1 Introduction

This section contains screenshots of results that were not considered vital to the understanding, nor evaluation of the program or report. It has been decided to enter excess results here in order to conserve space yet provide proof of working to varying specifications.

B.2 Minimum Criteria Results

B.2.1 Test One- Extras

As discussed in Testing Design, section 6.1.1, the ability to create buyer and seller arrays was tested first with array size six, then ten, then one hundred. The screenshot showing ten results resides in the main body of the report and it has been decided to place screenshots displaying six, figure B.1, and one hundred values here B.2. To display one hundred values the last value from each column in B.2 forms the start of the next.

B.2.2 Tests Four and Five- Extras

```
[scs5clf@cslin036 FYP]$ ./trader.py
buyer prices:
2.57
8.64
1.24
6.68
6.34
9.12
seller prices:
5.99
6.71
9.92
4.44
4.44
2.45
[scs5clf@cslin036 FYP]$
```

Figure B.1: Screenshot showing six randomly assigned values to each of the buyer and seller arrays.
Results from Test One

File	Edit	View	Terminal	Tab	File	Edit	View	Terminal	Tab	File	Edit	View	Terminal	Tab
[scs5clf@cslin036 FYP]\$./trader.py					[scs5clf@cslin036 FYP]\$./trader.py					[scs5clf@cslin036 FYP]\$./trader.py				
buyer prices:					buyer prices:					buyer prices:				
4.7	3.21	1.07	9.62	8.38	5.05	6.19	7.61	7.52	7.98	0.31	8.38	9.52	4.35	5.12
1.2	0.39	0.06	8.16	5.23	0.1	6.37	2.84	4.2	1.25	2.19	0.8	2.74	5.01	8.6
5.09	3.55	7.99	5.41	3.83	9.15	2.57	8.68	9.66	8.47	4.34	9.04	1.22	4.67	4.77
0.9	0.76	7.47	9.92	5.88	9.83	7.21	6.41	8.7	7.83	3.82	3.33	2.48	3.43	3.29
7.65	4.45													
seller prices:					seller prices:					seller prices:				
4.45	4.11	1.25	3.5	8.58	7.51	8.81	8.68	8.59	9.46	3.47	1.46	5.9	5.41	9.43
4.82	4.82	8.98	8.26	4.79	0.24	4.49	1.21	3.3	9.87	1.59	3.0	5.1	8.01	3.13
4.8	4.8	0.15	5.06	7.32	7.46	5.67	3.44	4.3	8.11	0.52	5.83	5.81	1.21	8.73
3.79	3.79	0.55	3.57	9.09	7.21	5.32	7.01	8.59	5.74	8.8	8.41	5.94	6.21	7.11
0.47	0.47	2.53												
[scs5clf@cslin036 FYP]\$					[scs5clf@cslin036 FYP]\$					[scs5clf@cslin036 FYP]\$				

Figure B.2: Screenshot showing one hundred randomly assigned values to each of the buyer and seller arrays. Results from Test One

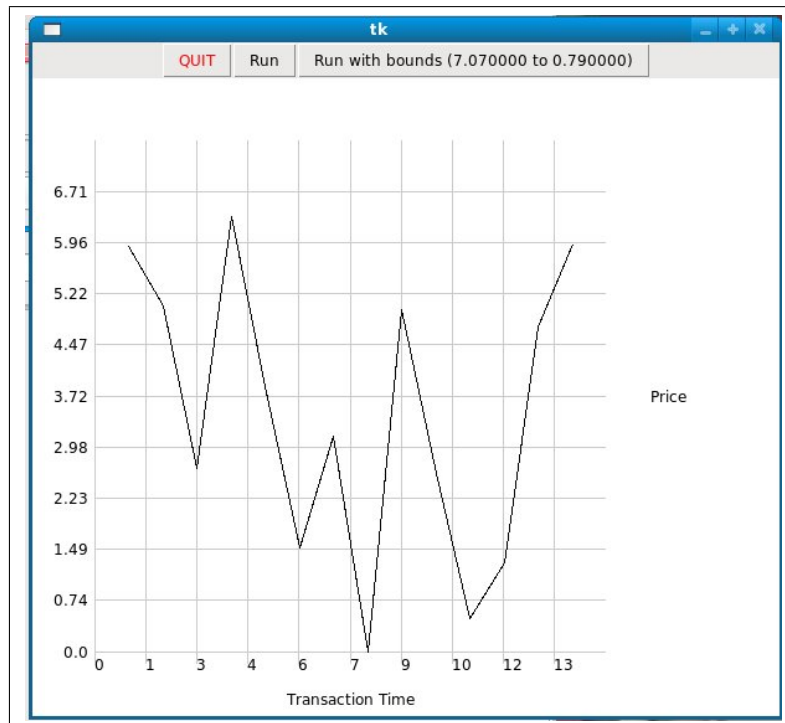


Figure B.3: Screenshot showing two days of trader activity. Results from Test Four

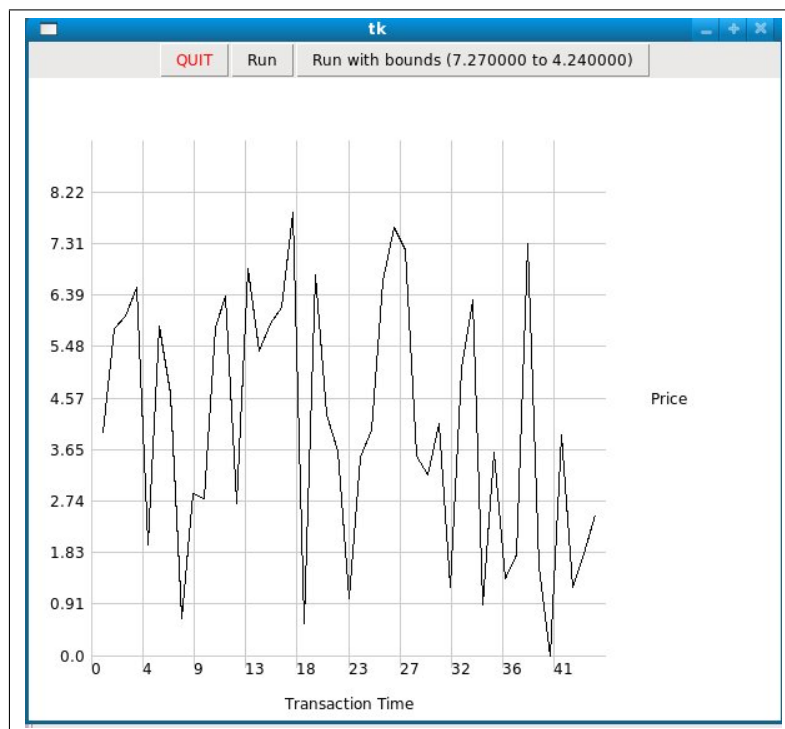


Figure B.4: Screenshot showing five days of trader activity, second run. Results from Test Four

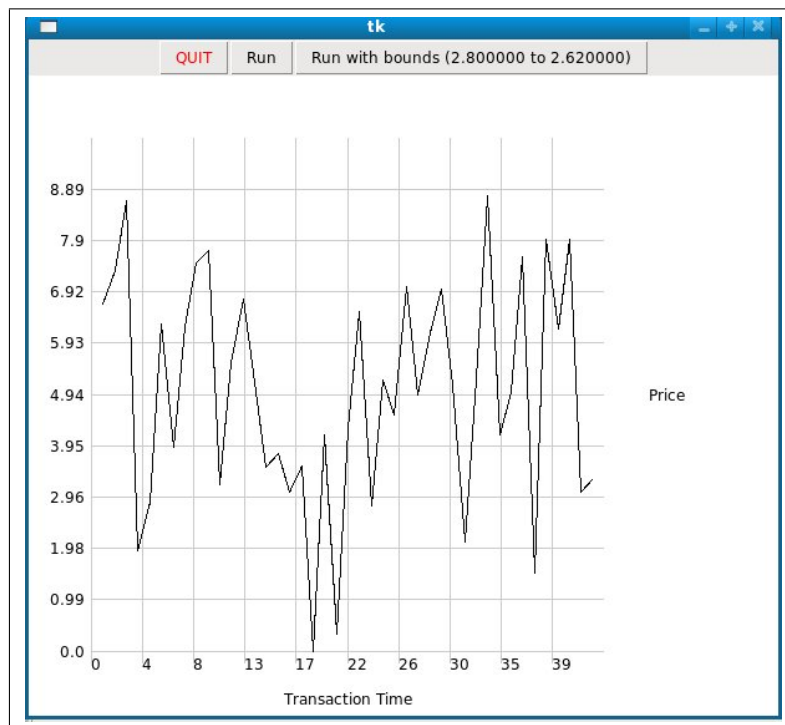


Figure B.5: Screenshot showing five days of trader activity, third run. Results from Test Four

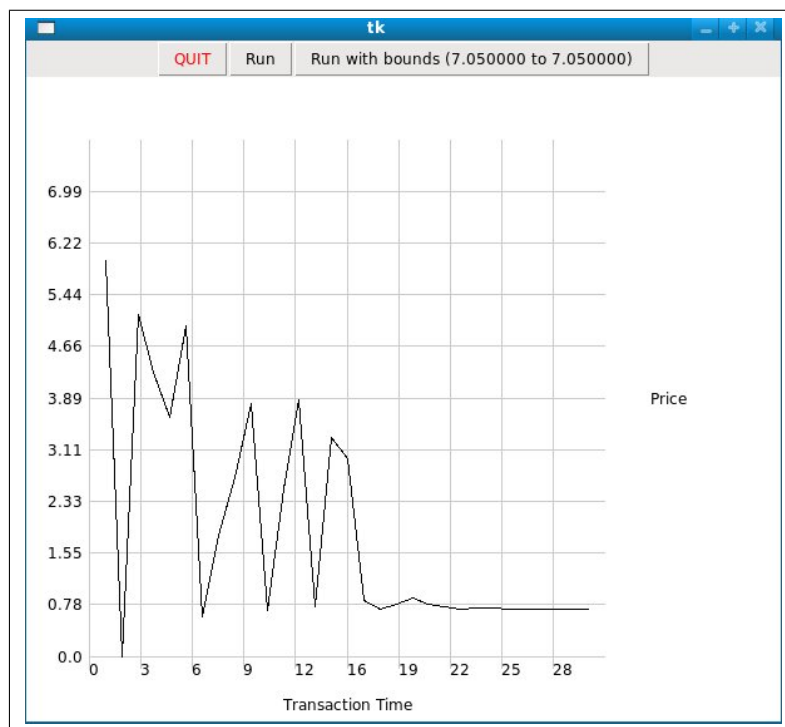


Figure B.6: Screenshot showing four, rather than the five required, days of ZIB trader activity, second run. Results from Test Five

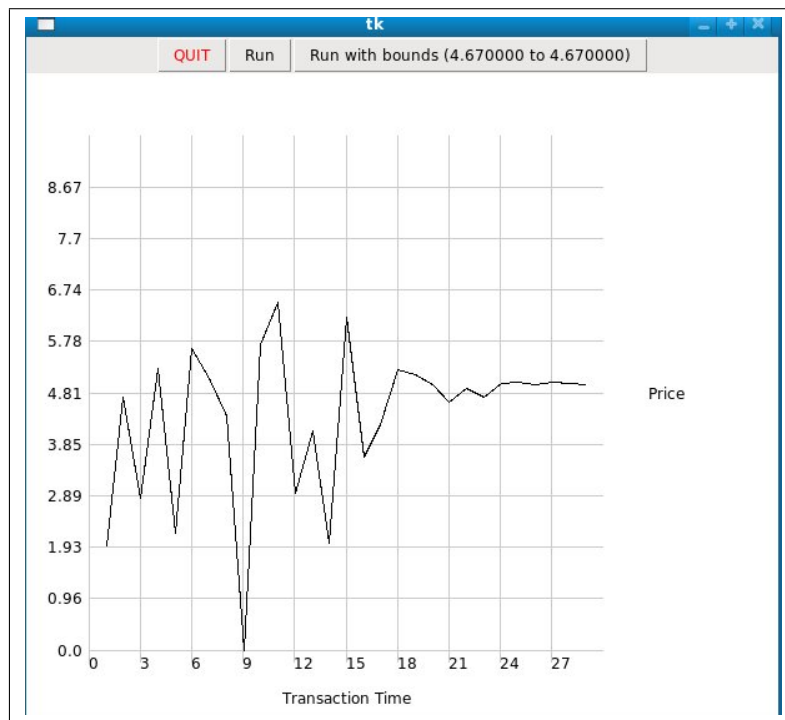


Figure B.7: Screenshot showing four, rather than the five required, days of ZIB trader activity, third run.
Results from Test Five

Appendix C

Abbreviation Reference List

Zero Intelligence Agents -ZI

Zero Intelligence Plus Agents -ZI-P

Zero Intelligence Agents with Constraints -ZI-C

Zero Intelligence Unconstrained Agents -ZI-U

Market Based Control -MBC

Graphical User Interface -GUI

Integrated Development Environment - IDE

Objected Orientated - OO

Appendix D

Appendix D- Timetables

Project Schedule			
Continuous deliverables- background research			
Aim	Start date	End Date	Work
Main Body of Background research	4 Nov	21 Dec	45 days
Architecture Design decided	3 Jan	10 Jan	7 days
Implement Buyer/Seller Arrays	24 Jan	12 Feb	19 days
Implement Bidding Mechanism	12 Feb	22 Feb	10 days
Implement Transaction GUI	22 Feb	4 Mar	10 days
Re-iterate to improve design	4 Mar	10 Mar	6 days
Implement improvements	10 Mar	18 Mar	8 days
Final Report	18 Mar	30 Apr	43 days

Figure D.1: Original Timetable from the Mid-Project Report

Project Schedule			
Continuous deliverables- background research			
Aim	Start date	End Date	Work
Main Body of Background research	4 Nov	21 Dec	45 days
Architecture Design decided	3 Jan	10 Jan	7 days
Implement Buyer/Seller Arrays	24 Jan	12 Feb	19 days
Implement Bidding Mechanism	12 Feb	22 Feb	10 days
Implement Transaction GUI	22 Feb	4 Mar	10 days
Re-iterate to improve design	4 Mar	10 Mar	6 days
Implement improvements	10 Mar	18 Mar	8 days
Final Report	18 Mar	30 Apr	43 days

Figure D.2: Updated Timetable

As can be seen, work halted around January/ February due to exams and each section took far longer than planned, especially the Background reading. This is described further in Appendix A.