Out of Equilibrium Dynamics with Decentralized Exchange Cautious Trading and Convergence to Efficiency

Sayantan Ghosal and James Porter

No 928

# WARWICK ECONOMIC RESEARCH PAPERS

**DEPARTMENT OF ECONOMICS**

THE UNIVERSITY OF WARWICK

# OUT-OF-EQUILIBRIUM DYNAMICS WITH DECENTRALIZED EXCHANGE: CAUTIOUS TRADING AND CONVERGENCE TO EFFICIENCY

Sayantan Ghosal and James Porter[1]

University of Warwick

March 19, 2010

Abstract

Is the result that equilibrium trading outcomes are efficient in markets without frictions robust to a scenario where agents' beliefs and plans aren't already aligned at their equilibrium values? In this paper, starting from a situation where agents' beliefs and plans aren't already aligned at their equilibrium values, we study whether out-of-equilibrium trading converges to efficient allocations. We show that out-of-equilibrium trading does converge with probability 1 to an efficient allocation even when traders have limited information and trade cautiously. In economies where preferences can be represented by Cobb-Douglass utility functions, we show, numerically, that the rate of convergence will be exponential. We show that experimentation leads to convergence in some examples where multilateral exchange is essential to achieve gains from trade. We prove that experimentation does converge with probability 1 to an efficient allocation and the speed of convergence remains exponential with Cobb-Douglass utility functions.

JEL Classification numbers: C62, C63, C78.

KEYWORDS: out-of-equilibrium, cautious, trading, efficiency, experimenting, computation

[1]We would like to thank Peter Hammond and Alan Kirman for their comments. Porter would like to thank EPSRC for PhD funding via the Complexity Science Doctoral Training Centre at the University of Warwick. Email: s.ghosal@warwick.ac.uk, j.a.porter@warwick.ac.uk

## 1. Introduction

Is the result that equilibrium trading outcomes are efficient in markets without frictions robust to a scenario where agents' beliefs and plans aren't already aligned at their equilibrium values? The first welfare theorem states that equilibrium allocations in competitive markets without frictions are efficient. In this paper, starting from a situation where agents' beliefs and plans aren't already aligned at their equilibrium values, we study whether out-of-equilibrium trading converges to efficient allocations.

We study a pure exchange economy with decentralized (pairwise, random matching) trading. With decentralized exchange, agents involved in a current match might be willing to take a utility loss (relative to their current holdings) in anticipation of a gain in a future match[2]. However, in our set-up, as agents are aware that their forecasts could be wrong, they behave cautiously and only carry out actions which improve their utility evaluated at their current holdings. Thus agents only propose or accept trades that improve their utility (relative to their current holdings) and they will do this with limited knowledge about the preferences of trading partner[3]. The resultant trading process is path dependent. We show that the process converges with probability 1 to a pair-wise optimal allocation which, under some additional on fundamentals, are also Pareto efficient. We show, via numerical simulation, the speed of convergence is exponential for a economies with Cobb-Douglass utility functions. Augmenting this process with experimentation leads to convergence in Scarf's example of an exchange economy with a unique globally unstable (under tatonnement dynamics) competitive equilibria where multilateral exchange is essential to achieve gains from trade. We characterize, analytically, the convergence properties of an augmented process with experimentation and show, numerically, that the speed of convergence remains exponential with Cobb-Douglass utility functions.

The approach adopted here contrasts with the classical one (see, for example (Arrow and Hahn 1971))- either with or without explicit trading in out-of-equilibrium scenarios (tatonnement or cobweb dynamics, non-tatonnement dynamics)- suffers from the problem that there is no explicit price formation rule nor are payoffs well-defined in out-of-equilibrium scenarios. The implication is that in contrast to equilibrium outcomes, the analysis of out-of-equilibrium scenarios depend on

[2]The interpretation is that random matching and bargaining models could be used to model exchange in primitive barter markets.

[3]Under the assumption that agents anticipate the outcomes of any future match correctly, a large literature has used models of decentralized exchange to study the strategic foundations of competitive equilibria (see for example (Rubinstein and Wolinsky 1985, Gale 1986*a*, Gale 1986*b*, McLennan and Sonnenschein 1991, Gale and Sabourian 2005)).

rules of price adjustment and allocation dynamics aren't grounded on the behaviour of agents. In this paper, in contrast, in our model decentralized exchange the map from action profiles to prices and allocations is well-defined both out-of-equilibrium and at equilibrium. Thus, out-of-equilibrium dynamics can be grounded explicitly on the (cautious) behaviour of agents.

Various attempts have been made to model trade in decentralised economies. Early results (Feldman 1973, Rader 1976) characterise the conditions required for for a decentralised bilateral exchange economy to converge to a Pareto Optimal allocation. (Goldman and Starr 1982) derives generalised versions of these results for $k$-lateral exchange where exchange happens between groups of $k$ agents. An alternative approach is the assumption of "zero intelligence" (Gode and Sunder 1993). Here there are a variety of computer agents, one form of which simply makes random offers subject to a budget constraint. They speculate that the "efficient" outcomes are due to the double auction market structure under investigation. Another angle is taken by Foley's work on statistical equilibrium, for example (Foley 1999), which models an economy via discrete flows of classes, that is homogeneous classes of, traders entering a market who have discrete sets of trades they wish to carry out. The result is probability distributions over trades, so as in our process agents with identical initial endowments may end up with different final allocations, but as in the many Walrasian frameworks, but unlike our approach, the trading process remains an unspecified black box. More recently (Gale 2000) has approached an out of equilibrium economy with a model with decentralized exchange in the special case with two commodities and quasi-linear utility functions.

(Axtell 2005) has explored decentralised exchange from a computational complexity perspective. He argues that the Walrasian auctioneer picture of exchange is not computationally feasible, while decentralised exchange is. While this adopts a somewhat decentralised (possibly bilateral perspective) it assumes a high level of information in the groups which are bargaining (essentially a Pareto optimal outcome for that group is directly calculated) and seems to sidestep the issue of coordinating the matching of these groups.

In a related contribution (Fisher 1981) studied a model of general equilibrium stability in which agents are aware they are not at equilibrium. In our paper, in contrast to (Fisher 1981) we do not require agents to hold their expectations with certainty and we allow for price setting by individual agents.

Gintis has looked at an agent-based model of both an exchange economy (Gintis 2006) and general equilibrium economy (Gintis 2007) although the dynamics in his models, driven by evolutionary selection, are limited to quite homogeneous agents (for example, in his exchange economy agents all have the same linear utility functions).

The remainder of the paper is structured as follows. The next section is devoted to the study of cautious trading. Section 3 presents numerical methods and results. Section 4 studies augmented processes with experimentation. The last section concludes. Appendix B presents the key sections of the source code.

## 2. Analytics

We consider individuals who are aware they are in an out-of-equilibrium state and thus realise they may make mistakes if they were to attempt to condition their current trade based on future expectations. In response to this we consider agents who only accept trades which improve upon their current holdings. We assume that the process is connected, that is at every time any given pair of agents will attempt exchange at some point in the future. This idea of accepting only improving trades in a connected exchange economy we call *cautious trading* and precisely specify below.

There are individuals $i \in I = \{1, \ldots, I\}$, commodities $j \in J = \{1, \ldots, J\}$ and endowments $e_i^j \in \mathbb{R}, e_i > 0$ of commodity $j$ for individual $i$. Trade takes place in periods $t \in 1, 2, \ldots$ and we write the bundle of commodities belonging to individual $i$ at time $t$ as $\mathbf{x}_{it}$ and restrict these to positive bundles (you can only trade what you currently have). Agents have strictly increasing real valued utility functions $u_i(\mathbf{x}_{it})$ which are defined for all non-negative consumption bundles[4].

In each period $t$ two agents are selected at random such that in any period there is an equal probability that any particular pair will be selected. We will assume that once a pair is matched the two agents put up all their current holdings for exchange; one agent, the proposer, which without loss of generality is $m$, proposes a non-positive[5] trade $\mathbf{z}_t$ to a responder $n$ such that:

$$x_{mt}^j > -z_t^j > -x_{nt}^j \quad \forall j$$

and

$$u_m(\mathbf{x}_{mt} + \mathbf{z}_t) > u_m(\mathbf{x}_{mt}).$$

The first condition is just that the trade would leave $m$ and $n$ with positive quantities of each good. The second condition is that the trade is utility increasing for $m$. The responder, $n$, will accept the trade if it weakly improves his utility, that is

$$u_n(\mathbf{x}_{nt} - \mathbf{z}_t) \geq u_n(\mathbf{x}_{nt}).$$

[4]Formally, the trading dynamics we study in this paper has the feature that agents do not consume till trade stops. However, following (Ghosal and Morelli 2004), note that a reinterpretation of our model so that agents trade durable goods that generate consumption flows within each period will allow for both consumption and trade.

[5]That is not simply proposing a gift: must be an exchange.

but reject it otherwise (in which case no trade takes place).

Note that the requirement that agents put up all their current holdings for exchange is without loss of generality. This is because no agent will have an incentive to conceal his holdings. First an agent is free to reject any offer that is put on the table. Secondly by concealing some of his holdings the agent reduces the probability of generating a mutually improving trade[6]. Therefore, our trading process requires that the proposer will need to know only his current holdings, his utility function and the responder's holdings.

Let us assume that the proposals are drawn at random from the set of all such proposer's utility improving proposals, $Z$, such that there is a strictly positive probability of choosing a proposal within any open set $X \subset Z$. Furthermore we will assume that the random choice of a new proposal will satisfy the following *minimal probability weight* condition: there exists some $c \in (0, 1]$ such that for all periods $t$ the probability of choosing a proposal from any open subset $X$ of $Z$ is greater than $cp$ where $p$ is the probability of choosing a proposal in $X$ if we choose from a multivariate uniform random distribution over $Z$.

This is a *connected trading process with cautious behaviour*, which we abbreviate to *cautious trading*.

An allocation $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ is *pairwise optimal* if there exists no way of redistributing bundles between any pair $m, n$ that would make at least one strictly better off, while making the other at least as well off. This notion can be generalised to $k-$wise optimality in the obvious way, see (Goldman and Starr 1982) for a full account of this concept and related results. If $k = I$ then we would be considering Pareto optimality.

**Proposition 1.** *The Cautious Trading process converges in utility and the allocations converge to a set of pairwise optimal utility-identical allocations.*

*Proof.* The sequence of utility values for any agent is increasing as only mutually utility increasing trades will be made. It is bounded as the set of feasible allocations is compact (the sum of all goods must be the sum of the endowments) and a maximum utility value for each agent is the value when it has all of all goods. So for each agent $i$ the sequence of utility values $u_i(\mathbf{x}_{it})$ converges to its supremum; call the vector of these $\bar{\mathbf{u}}$.

Now consider the sequence of allocations $\mathbf{X}_t$ generated by cautious trading. Any limit points of such a sequence must be pairwise optimal allocations with utilities $\bar{\mathbf{u}}$. Suppose it wasn't then by definition there would exist a pair of agents who could be made better of by a

[6]Analytically for the below convergence results we could work with a weaker condition, an upper bound, but this form is numerically convenient, something we will return to in section 3.

trade $\mathbf{z}$. But by assumption there is a strictly positive lower bound on the probability of picking a trade within every neighbourhood of $\mathbf{z}$ in every period. By continuity there exists some such neighbourhood which pairwise improves (there may in fact be additional regions of our allocation space where this holds) so we know that a trade will almost surely happen at some point in the future between these two agents and so this cannot be an allocation at the limit of utility. □

The following example makes clear the crucial role of the minimal probability weight condition in obtaining convergence to pairwise optimal allocations.

**Example.** *Suppose there are two agents $a$ and $b$. We will set up our example such that there is a non-zero probability that trade will never occur. Consider $a$'s proposals to $b$; assuming that no trade occurs the set these are drawn from will not depend on time. Furthermore we can partition the set of improving trades $Z$ into $Z_a$ the set of individually improving but not weakly improving to $b$ trades and $Z_{(a,b)}$ the set of mutually weakly improving trades. Assume we are not at a pairwise (in this example trivially Pareto) optimal allocation and that $Z_a$ is also non-empty. Now assume that $a$ draws its proposals from a fixed probability distribution for each proposal in this particular state. That is it picks a $z \in Z_a \cup Z_{(a,b)}$. Now let $p^a$ be the probability it picks a proposal in $Z_a$ and $p^{(a,b)}$ be the probability it picks a proposal in $Z_{(a,b)}$. It has been assumed that there is a strictly positive probability of choosing a proposal within any open set $X \subset Z$, so this applies in particular to $Z_a$ and $Z_{(a,b)}$.*

*Now consider a new process where we transform the probability distributions over the disjoint sets $Z_a$ and $Z_{(a,b)}$ by a constant scaling such that $p_t^a = (1-\tau_t)p^a$ and $p_t^{(a,b)} = \tau_t p^{(a,b)}$ where $\tau_t$ is given by the sequence $\tau_t = \frac{1}{2^{t+1}}$ for time periods $t = 1, 2, \ldots$. We make no restrictions on the behaviour if we were to leave the initial state and claim that there is now a positive probability that trade will never occur so a fortiori we will not converge to a pairwise/Pareto optimal.*

*To see this consider the probability of at some point proposing a trade in $Z_{(a,b)}$, that is one which will be accepted. This is strictly less than $p^{(a,b)} \sum_t \tau_t = \frac{p^{(a,b)}}{2}$, which implies there is a non-zero probability that trade will never occur. Actually to complete this argument we need $b$ to propose in the same way. If both agents are proposing in this fashion then there is a non-zero probability that trade, and hence any kind of convergence, will never occur. Note that is is possible to generalise this to a larger number of agents by using the same weights on each distribution of proposals of $a$ to any agent $c$.*

While this example is somewhat pathological it illustrates an important point. For cautious trade to work we can't have agents conditioning their actions on the period in a way which essentially rules out trade at all, or via a limiting process. [7]

The following proposition shows that under the assumptions made cautious trading will get arbitrarily close to the Pareto frontier in finite time.

**Proposition 2** (First Welfare Theorem for Cautious Trading)**.** *(i) If the utility functions are continuously differentiable on the interior of the consumption set a Pairwise optimal allocation is Pareto optimal. (ii) If indifference surfaces through the interior of the allocation set do not intersect the boundary of the allocation set then if one agent has some of all goods and others have some of at least one good then cautious trading converges to a Pareto optimal.*

*Proof.* Let $\mathbf{X}$ be a pairwise optimal allocation in the interior of the allocation set. If we are in the interior of the allocation set then by assumption marginal rates of substitution exist for each agent for each pair of goods. These must be equal for every pair of agents otherwise a pairwise improvement would be possible. So they must be equal for all agents which implies that the allocation $\mathbf{X}$ is Pareto optimal.

From proposition 1 we know that the sequence converges to a set of Pairwise optimal states, so under the extra conditions imposed above it converges to a set of Pareto optimal states.

Now consider the case where one agent has some of all goods and others have some of at least one good. We need to establish that the process reaches the interior of the the allocation set then the result follows by the above argument.

Consider an agent on the boundary of the allocation set. As the indifference curves through the interior of the allocation set for all agents do not intersect the boundary of the set it is always in the agent's interest to accept a trade away from the boundary. When paired with an agent with some of all goods the agent can propose a trade which leaves him with some of all goods should it take place. As utility functions are strictly increasing and continuous there exists an open set of such trades that would be utility improving to the agent with all goods. So eventually such at trade will happen. This argument trivially extends to all agents on the boundary, so in finite time we will reach an allocation in the interior of the allocation set. □

**Corollary 1.** *If after some finite time an exchange process begins cautious trading, then it will converge to a Pairwise/Pareto optimal allocation subject to above conditions.*

[7]Note that in this example we have assumed that agent $a$ needs to know the utility function of agent $b$. One could weaken this to an assumption of the knowledge of the forms of utility functions over an economy as a whole.

So we could have some kind of initial experimentation process or trading conditioned on future expectations based on empirical distribution of trades and still obtain the same result if eventually cautious trading commences.

2.1. **Extension to Production.** Our convergence results for exchange can be extended to economies with production in view of (Rader 1964, Rader 1976). Formally an exchange economy is an array $\{(u_i, \mathbf{e}_i, \mathbb{R}^J_+): i \in I\}$. An economy with production is an array $\{(u_i, \mathbf{e}_i, \mathbb{R}^J_+) : i \in I; (Y^f) : f \in F, \theta_{if} : f \in F, i \in I\}$ where $f \in F = \{1 \dots F\}$ is the set of firms and $\theta_{if}$ is individual $i$'s share in firm $f$ with $\sum_i \theta_{if} = 1, \forall f$. Assume that the production set $Y^f$ of firm $f$ is convex, non-empty, closed, satisfies the no free lunch condition ($Y^f \cap \mathbb{R}^J_+ \subset \{0\}$), allows for inaction (that is $0 \in Y^f$), satisfies free disposal and irreversibility (that is if $y \in Y^f$ and $y \neq 0$ then $-y \notin Y^f$). We can convert an economy with production to an economy with household production by endowing each individual $i$ with a production set $\tilde{Y}_i = \sum_f \theta_{if} Y^f$. Next, by using Rader's principle of equivalence, an economy with household production can be associated with an equivalent economy with pure exchange with indirect preferences defined on trades. The conditions under which pairwise optimality implies Pareto optimality with such indirect preferences follow directly from Theorem 2 and its applications in (Rader 1976).

## 3. Numerical Results

While we have shown that the sequence of allocations will converge to a Pareto optimal set, this does not answer the question of how long such a process will take to get close to Pareto optimal. This section examines this question via a numerical approach, showing that for a common class of utility functions, the average speed of convergence is, in a sense to be specified shortly, good[8].

Attention is focused on sets of heterogeneous agents with Cobb-Douglas preferences and random initial endowments as a benchmark case. We can represent the preferences by utility functions:

$$u_i(\mathbf{x_i}) = \sum_j \alpha_i^j \ln(x_i^j).$$

One can of course represent Cobb Douglas utilities by $u_i(\mathbf{x_i}) = \prod_j (x_i^j)^{\lambda_i^j}$. However, the logarithmic representation is preferred for numerical work because it has a considerably lower computational cost. We have initial

[8]This section has been written so as to be as accessible as possible to the non-programmer. Those with experience of programming may wish to skim this section, while consulting the source code directly, the key sections of which are included in appendix B.

endowments, $\mathbf{e}_i^j$ , of each commodity drawn from a uniform distribution over $(0, 1]$ and parameters $\alpha_i^j$ of the functions are again drawn from $(0, 1]$ uniformly, then normalised such that the sum, $\sum_j \alpha_i^j = 1$. They are normalised to a fixed value so as to make talking about global utility as the sum of agent's utilities more meaningful; this does not change the preferences which they represent.

As before trades are restricted to the set of all trades which leave both proposer $i$ and responder $j$ with positive quantities of each good, that is:

$$-x_{mt}^j < z_t^i < -x_{nt}^i$$

as to actually implement the trading process it is necessary to fix some boundary values[9]

The key "objects" we need in our computational model is an agent and a collection of agents. The former implements agents with Cobb-Douglas utility functions as specified above, random initial endowments and importantly specifies the actual mechanics of trade proposals, acceptance or rejection and trades. The later creates a collection of these agents and carries out realisations of the economy. A schematic representation of these classes can be found in figure 1. Utilising these we can obtain various numerical results via processes like that illustrated in figure 2.

We make one further major assumption: each agent makes one proposal per round, irrespective of the size of the economy. A natural way to approach implementing this model might be to fix some $n$, perhaps $n = 1$ as the total number of proposals per round, with agents drawn at random in each round. However, if we take seriously the decentralisation of the economy then we should assume that agents actions are unconstrained by the size of the global economy.

We looked mainly at estimated convergence in average global utility to assess the performance of cautious trading. To estimate this we calculate global utility by summing across utility for all agents in the economy, then take an average over many runs as the process is stochastic. One can then use the final value as an estimate of limiting utility and calculate how far away earlier values are. The last few hundred values are discarded as for them this estimate of limiting utility is not, relatively speaking, as good. The analysis depends on the increasing

[9]An alternative, and in some ways more satisfying alternative (as it limits required information), might be to restrict trades to within the total endowment of the economy. While analytically we would obtain the same asymptotic results, numerically it would simply lead to many rejected proposal and vastly longer running times if these were simulated directly. One could try and simulate the proposal process indirectly if one could formulate joint probability distributions over improving offers, over improving proposals and over agent pairing. However for anything other than trivial economies this is extremely difficult due to the number of dimensions and changing state when proposals accepted.

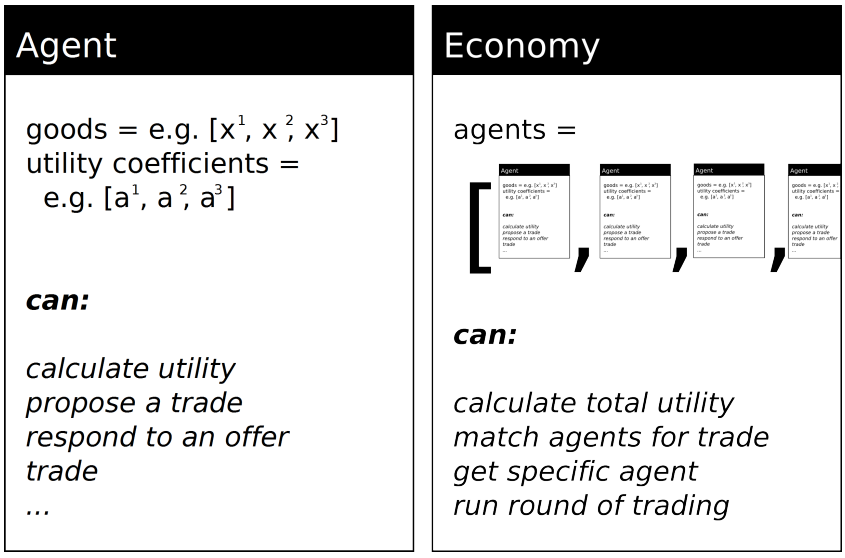


FIGURE 1. An outline of the main attributes and methods of the *Agent* and *Economy* objects.

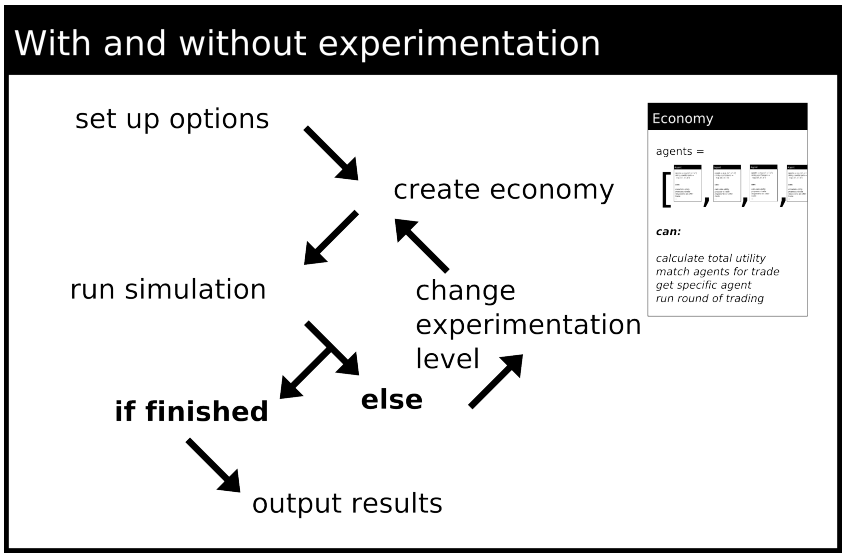


FIGURE 2. An example algorithm of a numerical simulation of Cautious Trading. The precise details vary depending the experiment being carried out but this example gives an overview of the kind of algorithm used to generate the data for most of the figures in this paper.

nature of sequences of utility values for agents this analysis to make sense.

In figure 3 one can see how varying the total number of agents effects the average speed of convergence. As one can see there is in fact very little qualitative effect. There is some increase in time taken, however

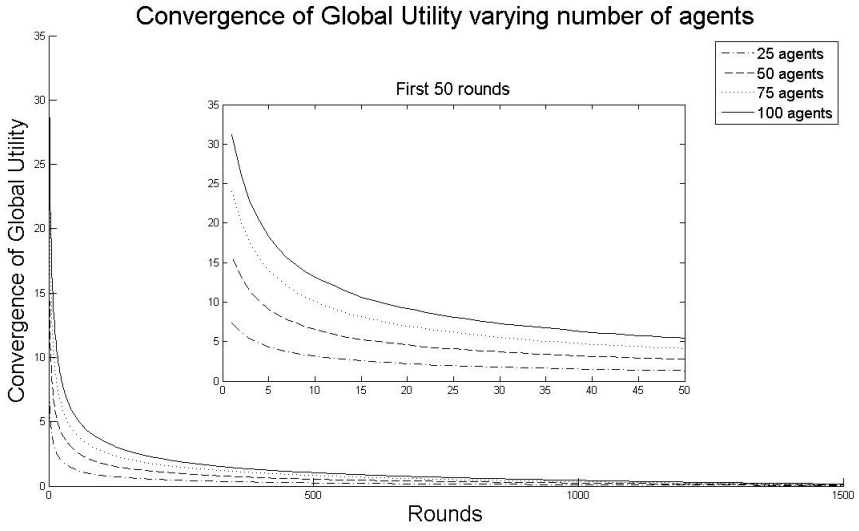


FIGURE 3. Average over many runs of global utility convergence when varying the total number of agents in the economy. *Parameters: 5 goods, 25-100 agents.*

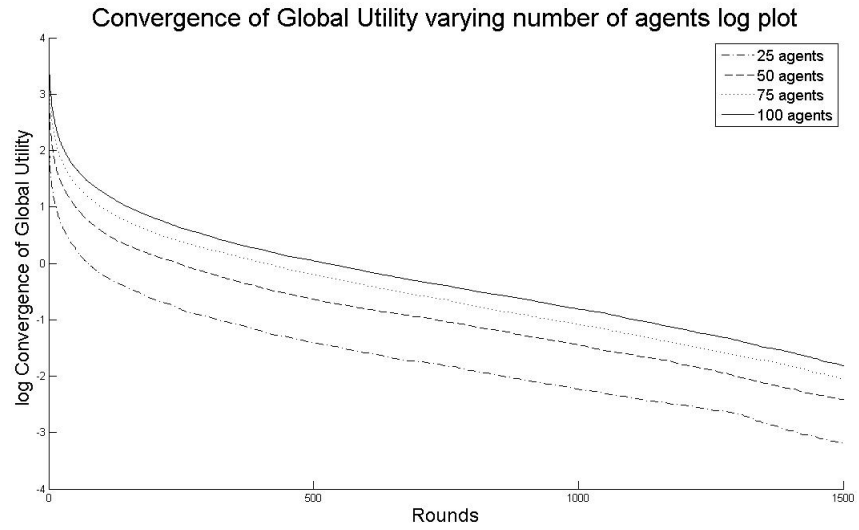


FIGURE 4. Log of average global utility convergence when varying the total number of agents in the economy. *Parameters: 5 goods, 25-100 agents.*

when one plots the log of average convergence as in figure 4 one can see that we get a close approximation to a straight line after an initial faster period; suggesting an exponential speed of convergence, at least over the these time periods.

We also examined the effect of the number of goods via similar analysis. In figure 5 one can see how the speed of convergence varies with

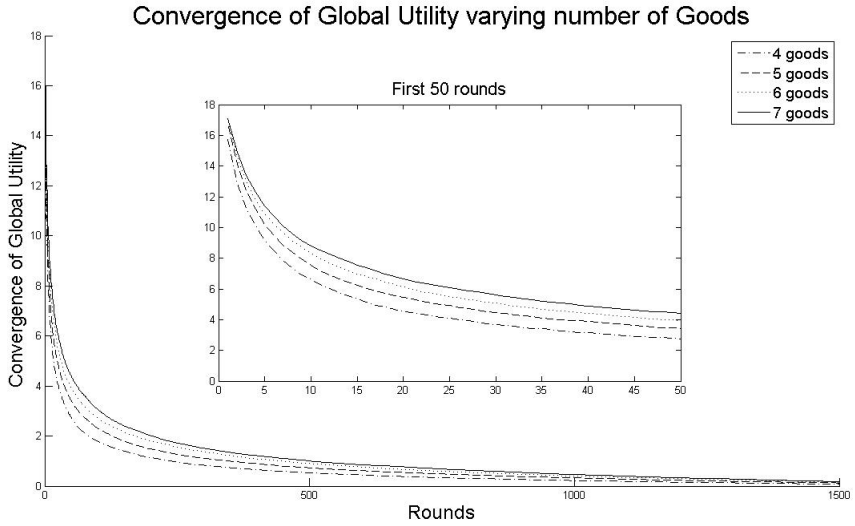


FIGURE 5. Average global utility convergence when varying the total number of goods in the economy. *Parameters: 4-7 goods, 50 agents.*

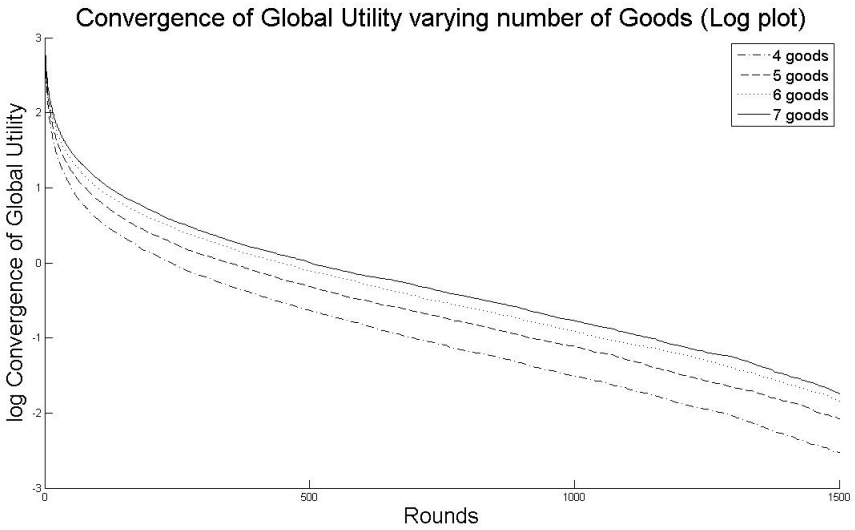


FIGURE 6. Log of average global utility convergence when varying the total number of goods in the economy. *Parameters: 4-7 goods, 50 agents.*

the total number of goods in the economy. There is a similar result of little qualitative change. This is more surprising as we have the same number of proposals taking place as before over larger increasing numbers of goods. When one examines the the log plot in figure 6 one gets the same kind of result as for varying agents.

| Goods | 4 | 5 | 6 | 7 |
|---|---|---|---|---|
| linear coefficient: | -0.0026 | -0.0019 | -0.0021 | -0.0021 |
| constant term: | 6.3642 | 5.6688 | 5.9079 | 6.3833 |

TABLE 1. Fitting exponential function to average convergence for varying numbers of goods. The values given are for linear fit of log of convergence. For every fit the $p$-values for the regressions are less than 0.0005 indicating an extremely high level of confidence in the fit of the model.

| Agents | 25 | 50 | 75 | 100 |
|---|---|---|---|---|
| linear coefficient: | -0.0023 | -0.0022 | -0.0022 | -0.0023 |
| constant term: | 4.4728 | 4.3750 | 5.3190 | 5.2299 |

TABLE 2. Fitting exponential function to average convergence for varying numbers of agents. The values given are for linear fit of log of convergence. For every fit the $p$-values for the regressions are less than 0.0005 indicating an extremely high level of confidence in the fit of the model.

One can fit an exponential function, via regression on the log of the values, to these average utility paths in order to obtain a numerical estimate for the average speed of convergence. In tables 1 and 2 we present such results for a range of model sizes. The important point to note is the approximately exponential convergence in global utility for a range of sizes of economy, both in terms of number of goods and number of agents, rather than the actual fitted parameters. Note that the $p$-values for the regressions are less than 0.0005 indicating an extremely high level of confidence in the fit of the model.

## 4. Experimentation

4.1. **Non-convergence of Cautious Trading.** Strong, though fairly standard, assumptions were required for the above analytical results obtained in section 2. However by introducing the idea of experimentation or making "mistakes" we may be able to do better. One famous class of examples that show non-convergence and instability in a global competitive equilibrium is presented in (Scarf 1959). This example can be adapted for our model in a similar way to (Gintis 2007): the basic idea is that there are three classes of agents each of whom has a utility function which is the minimum of the good it has and one other; but no agent, at least initially, can find an agent with whom a mutually

improving trade can take place. To specify precisely:

$$u_1 = \min(x^1, x^2) \text{ with endowment } e_1 = (1, 0, 0)$$
$$u_2 = \min(x^2, x^3) \text{ with endowment } e_2 = (0, 1, 0)$$
$$u_3 = \min(x^1, x^3) \text{ with endowment } e_3 = (0, 0, 1)$$

This means that in the Cautious Trading model, and similar models, no trade will ever take place. In figure 7 we take this example outlined above and examine what happens numerically, introducing a small probability $\epsilon$ of making a mistake, that is proposing or accepting a disimproving trade. If no experimentation takes place no trade ever happens and global utility remains at 0. As we increase the level of experimentation short term global utility improves (rises more steeply) at the cost of a lower level of long term convergence. In cautious trading form nothing happens, but with experimentation trade happens.

For high values of experimentation faster initial improvement than low values, but longer term global utility is slightly lower and the economy more volatile. This suggests that in selecting the level of experimentation there is a trade off between convergent level of utility and speed of convergence.

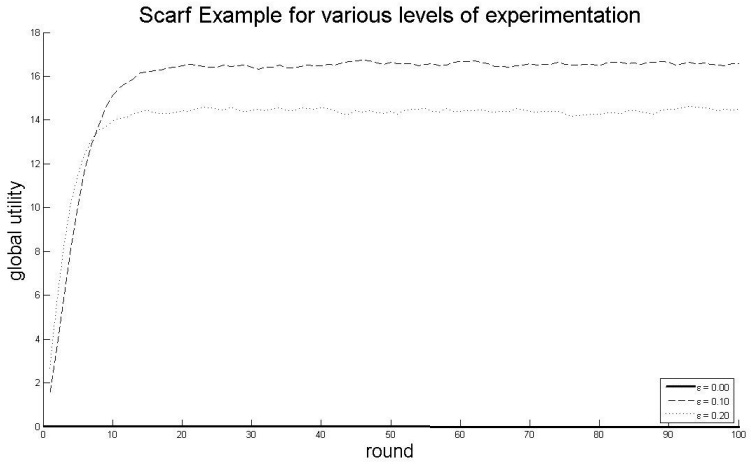


FIGURE 7. Without experimentation no trade takes place in this model adapted from a model of Scarf. Notice how long term utility appears lower for a higher level of experimentation.

4.2. **Analytical Model and Results for Experimentation.** To formally introduce experimentation we draw on the idea of simulated annealing (Kirkpatrick and Vecchi 1983), a search technique which has been used in a wide variety of computational search problems, with much practical success. The basic idea is that we augment a hill climbing search, something closely analogous to the basic form of cautious

trading, where we make small (bilateral trade) improving steps, with a temperature parameter which in some way determines the probability of making a disimproving step, that is going downhill on the search landscape. Over time the probability of making such moves decreases according to a cooling schedule and the algorithm will settle at some maxima the original algorithm may not have found or at least might not have found so quickly.

One can define the cautious trading with experimentation process as before for the proposer; however we now have some experimentation function $f$ from current period in $\mathbb{N}$ to a probability in $[0, 1]$ with limit 0 as $t \to \infty$. This is the probability in a given period that experimentation will take place. One can augment with a further function $h$ which determines the loss in utility that is acceptable in a given period (that is loss in utility for each agent engaged in trade), subject to a similar condition that the limit is 0, that is no loss is deemed acceptable at the limit.

*Total experimentation is almost surely finite* if the composite process described above leads to a total loss across all time to all agents that is bounded with probability 1. Any form of experimentation which ceases in finite time will trivially satisfy the above.

**Proposition 3.** *If total experimentation is almost surely finite then cautious trading augmented with experimentation converges with probability* 1 *to a set of Pairwise optimal allocations.*

*Proof.* For a particular realisation let $\mathbf{x}_i^t$ be the current allocation of agent $i$ at time $t$, $u_i^t$ the utility of agent $i$ at time $t$. Let $e_i^t$ be the loss in utility to agent $i$ in period $t$. If no experimentation occurs in period $t$ for agent $i$ then $\epsilon_i^t = 0$. By assumption the total amount of experimentation of all agents is almost surely finite, so for any particular agent the sum of $\epsilon_i^t$ is also finite.

Let the sequence $v_i$, indexed by $t$, be given by $v_i^t = u_i^t + \sum_{k=1}^{t} \epsilon^t$. Then this new sequence $v_i$ is increasing. It is also bounded as it is the sum of two bounded sequences. Therefore it converges to a limit, say $\tilde{v}_i$. But this implies that $u_i$ also converges to some limit $\tilde{u}_i$.

Now consider once more allocations at this limit $\tilde{u}_i$. They must be pairwise optimal as if they weren't then a pairwise improving trade would be made at some point in the future, even without experimentation. □

Even if we augment the trading process with the possibility that agents may trade to boundary allocations, subject to the conditions of continuity and strict monotonicity this will never occur under the conditions specified below.

**Proposition 4.** *If furthermore utility functions are continuously differentiable and indifference curves in the interior of the allocation set*

*do not intersect the boundary, the process of Cautious Trading augmented with experimentation will with probability* 1 *both*

(1) *not go to an allocation on the boundary*
(2) *and will converge to a set of Pareto Optimal allocations.*

*Proof.* To go to an allocation on the boundary with the above conditions an agent must in effect accept an infinite loss in utility; as we assume that the amount of experimentation is almost surely finite this will almost surely not occur.

If the utility functions are continuously differentiable then any pairwise optimal allocation is a Pareto optimal allocation as the marginal rates of substitution of goods for each agent must be equal. By proposition 3 the process converges to a set of pairwise optimal allocation allocations, so with the additional assumption this is Pareto optimal. □

4.3. **Experimentation and the speed of convergence.** Above an example adapted from Scarf was presented which showed how experimentation could lead to a better outcome than before, however, this example is a very special case. An interesting question we can ask numerically is how experimentation effects the speed of convergence in a larger, more heterogeneous example such as the Cobb Douglas utility function economy we looked at previously. In fact there is qualitatively similar long term behaviour when experimentation is included as can be seen in figure 6 where experimentation is introduced into the original model from section 3. For certain values of experimentation we even see slightly better overall performance with experimentation.

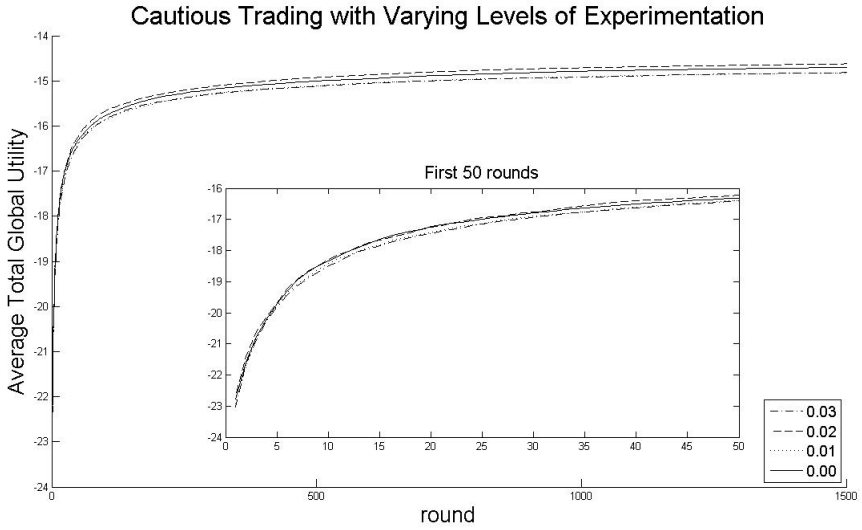


FIGURE 8. Low levels of experimentation have little effect in the Cobb Douglas economy we looked at before.

So we have seen that cautious trading allows trade to occur when it would not have otherwise happened. Furthermore, in economies where experimentation is not required, such as the Cobb-Douglas economy in figure 8, experimentation does not appear to have a qualitatively detrimental effect.

## 5. Conclusions

Even with "zero information" an exchange economy with typical assumptions will converge to a Pareto optimal outcome purely through bilateral exchange among uninformed partners. It is possible to numerically examine the speed of convergence which turns out to be exponential for a typical class of utility function. Augmenting this process with experimentation leads to both convergence in some examples where it did not previously occur and potentially faster convergence in cases which did converge previously.

One can conceive of this "zero information" as a worst case assumption. In "real" markets one presumably has more to work with but almost never the kind of complete information that is typically assumed in comparable models of exchange. The dual discipline of having to deal with decentralisation and its resultant lack of information (not simply uncertainty over a small number of possible states of the world) and having to explicitly implement the models for numerical investigation has proved useful. A possible next step would be to examine out-of-equilibrium dynamics in asset trades.

## References

Arrow, K. and F.H. Hahn (1971). *General Competitive Analysis.*

Axtell, R. (2005). 'The complexity of exchange'. *The Economic Journal* **115**(504), F193–F210.

Feldman, A. (1973). 'Bilateral trading processes, pairwise optimality, and pareto optimality'. *The Review of Economic Studies* **40**, 463–473.

Fisher, F. M. (1981). 'Stability, disequilibrium awareness, and the perception of new opportunities'. *Econometrica* **49**(2), 279–317.

Foley, D. K. (1999). 'Statistical equilibrium in economics: Method, interpretation, and an example'.

Gale, D. (1986*a*). 'Bargaining and competition part i: Characterization'. *Econometrica* **54**, 785–806.

Gale, D. (1986*b*). 'Bargaining and competition part ii: Existence'. *Econometrica* **54**, 807–818.

Gale, D. (2000). *Strategic Foundations of General Equilibrium.* Cambridge University Press.

Gale, D. and H. Sabourian (2005). 'Competition and complexity'. *Econometrica* **73**, 739–769.

Ghosal, S. and M. Morelli (2004). 'Retrading in market games'. *Journal of Economic Theory* **115**, 151–181.

Gintis, H. (2006). 'The emergence of a price system from decentralized bilateral exchange'. *Contributions to Theoretical Economics* **6**(1), 1302–1302.

Gintis, H. (2007). 'The dynamics of general equilibrium'. *Economic Journal* **117**(523), 1280–1309.

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

Goldman, S. M. and Ross M. Starr (1982). 'Pairwise, t-wise and pareto optimalities'. *Econometrica* **50**(3), 593–606.

Kirkpatrick, S. Gelatt, C. D. and M. P Vecchi (1983). 'Optimization by simulated annealing'. *Science* **220**, 671–680.

McLennan, A. and H. Sonnenschein (1991). 'Sequential bargaining as a noncooperative foundation for walrasian equilibrium'. *Econometrica* **59**, 1395–1424.

Rader, T. (1964). Edgeworth exchange and general economic equilibrium.

Rader, T. (1976). Pairwise optimality, multilateral optimality and efficiency, with and without externalities. In 'Economics of Externalities'. Academic Press.

Rubinstein, A. and A. Wolinsky (1985). 'Equilibrium in a market with sequential bargaining'. *Econometrica* **53**, 1133–1150.

Scarf, H. E. (1959). 'Some examples of global instability of the competitive equilibrium'.

## Appendix A. Numerical Implementation

The numerical model was implemented using the Java programming language. The implementation explicitly models individual agents via an Agent class. Each instance of the class stores the agent's current bundle of goods, the parameters of its utility function and its current marginal rates of substitution. Each agent can make or consider offers, carry out trades and reset itself for another realisation of trading. In figure 2 a schematic version of the type of algorithm used is presented.

The following subsections outline the details of the models and implementation[10]. The key source files are contained in appendix B.

A.1. **Cautious Trading.** Two files contain the key parts of the implementation of Cautious Trading: the *Agent* and *CautiousEconomy* classes. The former implements agents with Cobb-Douglas utility functions, random initial endowments and specifies the mechanics of trade proposals and trades. The later creates a collection of these agents and carries out simulated runs of the economy. An a schematic representation of these classes can be found in figure 1.

A.2. **Scarf Example.** A modified version of the *Agent* and *CautiousEconomy* classes was created to study the behaviour of an economy which in many settings may not converge. The implementation is broadly similar to the original, the main changes being to the endowments and utility functions.

[10]Source code is available from `http://www.warwick.ac.uk/go/jamesporter` which includes all the examples used in this paper, along with code for other numerical experiments.

A.3. **Experimentation.** The *CautiousEconomy* has been augmented with the possibility of experimentation. Essentially the *ExperimentingEconomy* class adds experimentation to *CautiousEconomy* via a scaling parameter to proposed trades. To be more precise an initial level of allowable experimentation is selected and the allowable level decreases linearly until it ceases. The probability of experimentation is fixed at an initial level and this too decreases over time.

## Appendix B. Source Code

This section contains the key source code files; many more were actually used to model the cautious economy. The code[11] is arrange into four distinct levels: agent, economy, experiment and simulation. The first two play obvious roles, the experiment code provides general code to investigate the cautious economy and the simulation code runs experiments and does some processing of results. Figures in this report were then produced using Matlab.

B.1. **Agent Code.** The below code is for the basic form of the Agent class.

Listing 1. Agent class source code

```
1  package com.porter.cautious.model;
2
3  import java.util.Random;
4
5  public class Agent {
6
7    protected double goods[];
8    protected double originalGoods[];
9    protected double exponents[];
10   protected double originalExponents[];
11   protected double currentUtility;
12   protected int nGoods;
13   protected Random gen;
14
15   public Agent(int nGoods){
16     this.nGoods = nGoods;
17     gen = new Random();
18
19     goods = new double[nGoods];
20     exponents = new double[nGoods];
21
22     originalGoods = new double[nGoods]; //initial endowment
           stored for restart
23     originalExponents = new double[nGoods]; //initial
         exponents stored for restart
24       initializeRandomly();//actually initialise these
           arrays
```

[11] As mentioned previously, see `http://www.warwick.ac.uk/go/jamesporter`

```
25
26      update();
27    }
28
29    /**
30    * Initialise the agent with a random set of exponents and
          goods; then normalising the exponenents
31    */
32
33    protected void initializeRandomly() {
34        for(int i=0; i < nGoods; i++){
35            goods[i] = gen.nextDouble();
36            originalGoods[i] = goods[i];
37
38            exponents[i] = gen.nextDouble();
39            originalExponents[i] = exponents[i];
40        }
41        normalise();
42    }
43
44    /**
45     * Reset the agent, i.e. generate new endowments and
          exponents
46     */
47    public void reset() {
48        initializeRandomly();
49      update();
50    }
51
52    /**
53     * Restart the agent, i.e. restore endowments and
          exponents
54     */
55    public void restart() {
56      restore();
57        update();
58    }
59
60    /**
61    * Update utility when this is necessary. Should add any
        other update
62    * actions here.
63    */
64    protected void update(){
65      updateUtility();
66    }
67
68    /**
69     * Restore original state of agent
70     */
71    protected void restore() {
72        for(int i=0; i < nGoods; i++){
```

```
73          goods[i] = originalGoods[i];
74          exponents[i] = originalExponents[i];
75        }
76        normalise();
77    }
78
79    /**
80       * Normalise the utility function so exponents sum to
             1.
81       */
82    protected void normalise() {
83      double sum = 0.0;
84      for(int i=0; i < goods.length; i++){
85        sum += exponents[i];
86      }
87      assert sum != 0.0;
88      for(int i=0; i < goods.length; i++){
89        exponents[i] = exponents[i]/sum;
90
91      }
92    }
93
94    /**
95       * Return utility of an agent (Cobb-Douglas)
96       */
97    public double utility() {
98      return utility(this.goods);
99    }
100
101   /**
102      * Assess the utility of bundle (Cobb-Douglas)
103      */
104   public double utility(double[] bundle) {
105     double u = 0.0;
106     for(int i = 0; i < goods.length; i++){
107       u += (Math.log(bundle[i]) * exponents[i]);
108     }
109     assert(! Double.isNaN(u));
110     return u;
111   }
112
113   /**
114   * Assess the utility if give a (i.e. subtract a) bundle.
115   */
116   protected double utilityIfGive(double change[]) {
117     double[] temp = new double[nGoods];
118     for(int i=0; i<nGoods; i++){
119       temp[i] = this.goods[i] - change[i];
120     }
121     return utility(temp);
122   }
123
```

```
124   /**
125   * Assess the utility if get a (i.e. add a) bundle.
126   */
127   protected double utilityIfGet(double change[]) {
128     double[] temp = new double[nGoods];
129     for(int i=0; i<nGoods; i++){
130       temp[i] = this.goods[i] + change[i];
131     }
132     return utility(temp);
133   }
134
135   /**
136      * Update the current utility level - call this if you
           change the bundle or exponents
137      */
138   protected void updateUtility() {
139           currentUtility = utility();
140   }
141
142   /**
143        * An agent makes a proposal to another Agent other.
144        * @param The agent to propose offer to
145        * @return Whether a trade took place
146        */
147   public boolean propose(Agent other) {
148     double proposal[] = getProposal(other);
149
150     if (other.consider(proposal)){
151       trade(other, proposal);
152       return true;
153     }
154     else{
155       return false;
156     }
157   }
158
159   /** Consider a trade of change, return true if improving,
        false otherwise*/
160   public boolean consider(double change[]) {
161     if(utilityIfGet(change) > currentUtility){
162       return true;
163     }
164     else{
165       return false;
166     }
167   }
168
169   public boolean propose(Agent other, double
        allowable_experimentation) {
170     double[] proposal = getProposal(other);
171     if (other.consider(proposal, allowable_experimentation)
          ){
```

```
172       trade(other, proposal);
173       return true;
174     }
175     else{
176       return false;
177     }
178   }
179
180   protected double[] getProposal(Agent other){
181     double proposal[] = new double[nGoods];
182     boolean improving = false;
183     int j = 0;
184     while(!improving){
185       for (int i = 0; i < nGoods; i++) {
186         proposal[i] =  goods[i] - gen.nextDouble()*(goods[i
              ] + other.goods[i]);
187         assert(proposal[i] < goods[i]);
188       }
189       if(utilityIfGive(proposal) > currentUtility){
190         improving = true;
191       }
192       j++;
193     }
194     return proposal;
195   }
196
197
198   /**
199    * Consider a trade of change, return true if improving,
          false otherwise
200    */
201   public boolean consider(double change[], double
        allowable_experimentation) {
202     if(utilityIfGet(change) > currentUtility -
          allowable_experimentation){
203       return true;
204     }
205     else{
206       return false;
207     }
208   }
209
210   /**
211    * Agent gets the bundle of goods change (some or all
          components may be negative i.e. they lose this)
212    */
213   public void get(double change[]) {
214     for (int i = 0; i < nGoods; i++) {
215       goods[i] += change[i];
216     }
217     update();
218   }
```

```
219
220   /**
221    * Agent gives the bundle of goods change (some or all
          components may be negative i.e. they gain this)
222    */
223   public void give(double change[]) {
224     for (int i = 0; i < nGoods; i++) {
225       goods[i] -= change[i];
226     }
227     update();
228   }
229
230   /**
231    * Trading procedure: parameters: another Agent other and
          the trade to take place change.
232    */
233   public void trade(Agent other, double change[]) {
234     give(change);
235     other.get(change);
236   }
237
238   /** The exponents of the agent are shocked via a
        normalised
239   * Gaussian scaled via the shockSize parameter
240   * @param shockSize The scaling to be applied to a
        normalised Gaussian
241   * */
242   public void shockGaussian(double shockSize) {
243     for (int i = 0; i < nGoods; i++) {
244       exponents[i] += this.gen.nextGaussian()*shockSize;
245     }
246   }
247 }
```

B.2. **Cautious Economy Code.** The below code presents the basic economy class. All other version are built on this.

LISTING 2. Cautious Economy source code

```
1  package com.porter.cautious.model;
2  import java.util.*;
3  import java.io.*;
4
5  /** The class Economy consists of a collection
6   * of independent Agents, who trade
7   * via Cautious Trading.
8   */
9  public class CautiousEconomy {
10   List<Agent> agents;
11   public int size;
12   public int nGoods;
13   Random gen;
14   public int trades, period, round;
```

```
15
16  /**An Economy is of size no. of agents each
17   * of whom deal with nGoods no. of Goods.
18   * @param size Number of agents in economy
19   * @param nGoods Number of goods in economy*/
20  public CautiousEconomy(int size,int nGoods){
21    agents = new ArrayList<Agent>(size);
22    this.size = size;
23    this.nGoods = nGoods;
24    Agent a;
25    for (int i = 0; i < size; i++) {
26      a = new Agent(nGoods);
27      agents.add(a);
28    }
29    gen = new Random();
30    resetCounters();
31  }
32
33  /** Reset the economy i.e. give each agent a random
34   * allocation and utility function.
35   */
36  public void reset(){
37    for (Agent a: agents) {
38      a.reset();
39    }
40    resetCounters();
41  }
42
43  /**
44     * Restore original state of economy
45     */
46    public void restart(){
47    for (Agent a: agents) {
48      a.restart();
49    }
50    resetCounters();
51  }
52
53    protected void resetCounters(){
54        trades = 0;
55        period = 0;
56        round = 0;
57    }
58
59  /**
60     * Return the total utility of all Agents in the
          Economy.
61     */
62  public double totalUtility(){
63    double total = 0;
64    for (int i = 0; i < agents.size(); i++) {
65      total += agents.get(i).currentUtility;
```

```
66      }
67      return total;
68    }
69
70    /**
71      * Attempt one exchange per member of the economy
72    */
73    public void round(){
74              int r;
75              for (int i = 0; i < size; i++) {
76                  //Find another agent to propose to
77                r = gen.nextInt(size);
78                  while(r == i){
79                      r = gen.nextInt(size);
80                  }
81
82                  if(agents.get(i).propose(agents.get(r))){
83                      trades++;
84         }
85         period++;
86           }
87         round++;
88    }
89
90    /**
91       * Do many rounds of trading
92       * @param rounds no. of rounds of trading to carry out
93       */
94      public void iterate(int rounds){
95          for (int i = 0; i<size; i++) {
96              round();
97          }
98      }
99
100   /**
101    * Carry out multiple rounds of trading
102    * @param r Number of rounds to run
103    */
104   public void runRounds(int r){
105     for (int i = 0; i < r; i++) {
106       round();
107     }
108   }
109
110     protected void outputTotalUtility(FileWriter writer){
111         try{
112             writer.write("Total Utility: " + totalUtility()
                );
113         }
114         catch(IOException e){
115             e.printStackTrace();
116         }
```

```
117     }
118
119   /**
120    * @param periods The number of periods
121    * @param repetitions The number of realisations to
           average over
122    * @throws IOException
123    * */
124   public double[] averageUtility(int periods, int
          repetitions){
125     double results[] = new double[periods];
126     for (int i = 0; i < results.length; i++) {
127       results[i] = 0;
128     }
129
130     for (int r = 0; r < repetitions; r++) {
131       for (int i = 0; i < periods; i++) {
132         round();
133         results[i]+=totalUtility();
134       }
135       restart();
136     }
137     for (int i = 0; i < results.length; i++) {
138       results[i] /= repetitions;
139     }
140     return results;
141   }
142
143   /**
144    * @param periods The number of periods
145    * @param repetitions The number of realisations to
           average over
146    * @throws IOException
147    * */
148   public double[][] ManyUtility(int rounds, int repetitions
          ){
149     double results[][] = new double[rounds][repetitions];
150     for (int i = 0; i < rounds; i++) {
151       for (int j = 0; j < repetitions; j++) {
152       results[i][j] = 0;
153       }
154     }
155     for (int r = 0; r < repetitions; r++) {
156       for (int i = 0; i < rounds; i++) {
157         round();
158         results[i][r] = totalUtility();
159       }
160       reset();
161     }
162     return results;
163   }
164
```

```
165     public double[][] manyUtilityFixedIC(int rounds, int
            repetitions){
166         double results[][] = new double[rounds][repetitions
                ];
167
168         for (int i = 0; i < rounds; i++) {
169       for (int j = 0; j < repetitions; j++) {
170                     results[i][j] = 0;
171                 }
172              }
173         for (int r = 0; r < repetitions; r++) {
174       for (int i = 0; i < rounds; i++) {
175                 round();
176         results[i][r] = totalUtility();
177       }
178       restart();
179         }
180     return results;
181   }
182
183     /**
184      * Measure the rate of success of Cautious Trading.
185      * @param rounds
186      * @param intervals
187      * @return An array of the numbers of trades taking
             place in a series of intervals
188      */
189   public int[] measureRateOfSuccess(int rounds, int
         intervals){
190     int results[] = new int[rounds/intervals];
191     results[0] = 0;
192     int tradesSoFar;
193
194     for (int p = 0; p < rounds/intervals; p++) {
195       tradesSoFar = trades;
196       for (int i = 0; i < intervals; i++) {
197         round();
198       }
199       results[p] = trades - tradesSoFar;
200     }
201     return results;
202   }
203 }
```

B.3. **Scarf Example Source Code.** The below code is an adapted from Scarf's example. It takes a very simple form of experimentation and simple three class economy which does not converge via cautious trading and shows that experimentation will ensure trade and convergence in utility takes place. The code below presents only the code for the Agent class as the other code is roughly as before.

LISTING 3. Cautious Economy source code

```
1  package com.porter.cautious.scarf;
2
3  import java.util.Random;
4
5  import com.porter.util.Processing;
6
7  public class Agent {
8    protected int type, alsoDesired;
9    protected double goods[] = new double[3];
10   protected double currentUtility;
11   protected Random random;
12
13   /**
14    * Scarf Example type agent
15    * @param type 0,1,2 depending on good desired (in
          addition to endowed good)   */
16   public Agent(int type){
17     this.type = type;
18     this.alsoDesired = (type + 1) % 3;
19
20     random = new Random();
21     reset();
22   }
23
24   public void reset(){
25     switch(this.type){
26     case 0:
27       goods[0] = 1.0;
28       goods[1] = 0.0;
29       goods[2] = 0.0;
30       break;
31     case 1:
32       goods[0] = 0.0;
33       goods[1] = 1.0;
34       goods[2] = 0.0;
35       break;
36     case 2:
37       goods[0] = 0.0;
38       goods[1] = 0.0;
39       goods[2] = 1.0;
40     }
41     currentUtility = utility();
42   }
43
44   public double getUtility(){
45     return currentUtility;
46   }
47
48     protected double utility(){
49     return utility(goods);
50   }
```

```
51
52    protected double utility(double[] goods){
53      return Processing.min(goods[type],goods[alsoDesired]);
54    }
55
56    protected double utilityIfSubtract(double[] c){
57      double[] new_goods = new double[goods.length];
58      for (int i = 0; i < goods.length; i++) {
59        new_goods[i] = goods[i] - c[i];
60      }
61      return utility(new_goods);
62    }
63
64    protected double utilityIfAdd(double[] c){
65      double[] new_goods = new double[goods.length];
66      for (int i = 0; i < goods.length; i++) {
67        new_goods[i] = goods[i] + c[i];
68      }
69      return utility(new_goods);
70    }
71
72    public boolean propose(Agent other, double epsilon){
73      double proposal[] = new double[3];
74      boolean improving = false;
75      //experiment: propose at random
76      if(random.nextDouble() < epsilon){
77        for (int i = 0; i < 3; i++) {
78          proposal[i] = - goods[i] + random.nextDouble()*(
                goods[i] + other.goods[i]);
79        }
80      //don't experiment: propose an improving trade if that
           is possible
81      }else{
82        if(isImprovingTradePossible(other)){
83          while(!improving){
84            proposal = generateProposal(other);
85            if(utilityIfAdd(proposal) > currentUtility){
86              improving = true;
87            }
88          }
89        }else{
90          return false; //can't possibly benefit from trade;
               immediately return false without attempting
               trade
91        }
92      }
93      if(other.consider(proposal,epsilon)){
94        trade(other, proposal);
95        return true;
96      }else{
97        return false;
98      }
```

```
99  }
100
101 protected double[] generateProposal(Agent other){
102   double[] proposal = new double[3];
103   for (int i = 0; i < 3; i++) {
104     if (i != mostPressingNeed()){
105       proposal[i] = - goods[i] + random.nextDouble()*(
              goods[i] + other.goods[i]);
106     }
107     else{ //optimisation: negative proposal for this
            certainly not improving
108       proposal[i] = random.nextDouble()*other.goods[i];
109     }
110   }
111   return proposal;
112 }
113
114 protected int mostPressingNeed(){
115   if(goods[type] > goods[alsoDesired]){
116     return alsoDesired;
117   }
118   else{
119     return type;
120   }
121 }
122
123 protected boolean isImprovingTradePossible(Agent other){
124   if(goods[type] > goods[alsoDesired]){
125     return other.goods[alsoDesired] > 0.0;
126   }else{
127     return other.goods[type] > 0.0;
128   }
129 }
130
131 public boolean consider(double[] g, double epsilon){
132   if(random.nextDouble() < epsilon){
133     return true;
134   }else{
135     if(utilityIfSubtract(g) > currentUtility){
136       return true;
137     }
138     else{
139       return false;
140     }
141   }
142 }
143
144 public void update(){
145   currentUtility = utility();
146 }
147
148 public void trade(Agent other, double change[]){
```

```
149       addGoods(change);
150       other.subtractGoods(change);
151     }
152
153     public void addGoods(double[] g){
154       for (int i = 0; i < goods.length; i++) {
155         goods[i] += g[i];
156       }
157       update();
158     }
159
160     public void subtractGoods(double[] g){
161       for (int i = 0; i < goods.length; i++) {
162         goods[i] -= g[i];
163       }
164       update();
165     }
166
167
168 }
```

B.4. **Experimenting Economy Code.** The below code shows how the above economy has been expanded to include the idea of experimentation. We were able to utilise much of the functionality of the CautiousEconomy superclass. The key changes are to the round method and to the counters which are now of type double for efficiency purposes as we would otherwise need to cast integers to doubles to calculate experimentation scaling in each round.

LISTING 4. Experimenting Economy source code

```
1  package com.porter.cautious.model;
2
3  /** The class Economy consists of a collection
4   * of independent Agents, who trade
5   * via Cautious Trading.*/
6  public class ExperimentingEconomy extends CautiousEconomy {
7    public double acceptable, propensity, decay;
8    public double doubleEndDecay, doubleRoundCount;
9    public double baselineExperimentation;
10   public int endDecay;
11
12   /**An Economy is of size no. of agents each
13    * of whom deal with nGoods no. of Goods.
14    * @param size Number of agents in economy
15    * @param nGoods Number of goods in economy
16    * @param acceptable_loss_proportion The proportion of
          average initial
17    * absolute utility that is initially
18    * acceptable to lose in a trade.
19    * This declines until 0 at endDecay. Obviously there are
          schemes which are
```

```
20    * more analytically satisfying, this one is a compromise
          between this and ease of computation.
21    * @param propensity_to_experiment How often to
          experiment
22    * @param endDecay The point at which experimentation
          stops
23    * */
24   public ExperimentingEconomy(int size,int nGoods, double
          acceptable_loss_proportion, double
          propensity_to_experiment, int endDecay)
25           throws IllegalArgumentException{
26
27           super(size,nGoods);
28           //Check values of parameters
29           if ( 0.0 > acceptable || acceptable > 1.0
30       || 0.0 > propensity_to_experiment ||
             propensity_to_experiment > 1.0
31       || 0 > endDecay){
32         throw new IllegalArgumentException("Values must be in
               range (0,1] for Acceptable, Propensity and
               positive integer for endDecay");
33           }
34
35           this.propensity = propensity_to_experiment;
36           this.endDecay = endDecay;
37           this.doubleEndDecay = (float) endDecay;
38
39           this.baselineExperimentation =
               calculateBaselineExperimentation(
               acceptable_loss_proportion);
40   }
41
42   /**
43    * No experimentation version of Economy, should perform
          as Cautious Economy
44    * @param size
45    * @param nGoods
46    * @param acceptable
47    * @param proportion
48    * @throws IllegalArgumentException
49    */
50   public ExperimentingEconomy(int size,int nGoods) throws
          IllegalArgumentException{
51     this(size, nGoods, 0.0,1.0, 0);
52   }
53
54   protected double calculateBaselineExperimentation(double
          acceptable_loss_proportion){
55     return acceptable_loss_proportion *
           calculateAverageAbsoluteUtility();
56   }
57
```

```
58    protected double calculateAverageAbsoluteUtility(){
59      double total = 0.0;
60      for (int i = 0; i < size; i++) {
61        total += Math.abs(agents.get(i).currentUtility);
62      }
63      return total /(double)size;
64    }
65
66      @Override
67    public void round(){
68      double allowable_experimentation = this.
            baseline Experimentation *
69                        (1.0 - doubleRoundCount/
                              doubleEndDecay);
70          int r;
71
72          for (int i = 0; i < size; i++) {
73            r = gen.nextInt(size);
74
75            //get another agent at random
76            while(r == i){
77                  r = gen.nextInt(size);
78              }
79
80              if(this.round < endDecay && gen.nextDouble() <
                    propensity * (1.0 - doubleRoundCount/
                    doubleEndDecay)){
81              if(agents.get(i).propose(agents.get(r),
                    allowable_experimentation)){
82                trades++;
83              }
84              }
85            else{
86              if(agents.get(i).propose(agents.get(r))){
87                  trades++;
88                }
89            }
90            period++;
91          }
92          round++;
93          doubleRoundCount++;
94      }
95
96    @Override
97    protected void resetCounters(){
98      super.resetCounters();
99          doubleRoundCount = 0.0f;
100     doubleEndDecay = 0.0f;
101   }
102 }
```