PRINCETON UNIVERSITY

Expanding EOS

Developing and Understanding a Stable Baseline for the Economics by Object-oriented Simulation Framework

Michael Adelson '11

Submitted for fall semester independent work.

Faculty Advisor:

Kenneth Steiglitz

1/4/2010

Honor Code

This paper represents my own work in accordance with university regulations.

Michael Adelson

Abstract

EOS is a framework and computational laboratory for agent-based computational economics developed in spring 2009 by Chris Rucinski '10, Ye "Cody" Wang '10, and myself under the guidance of Professor Steiglitz. Here, we build upon this earlier EOS work by developing a stable baseline model and accompanying agent implementations as well as an I/O package to support both development and experimentation. The steps leading up to, and the analysis motivating the final "baseline 3" model are described, and the model is tested with a set of simple experiments. Finally, the baseline 3 model is compared to an agent-based economic model created and presented by Oeffner [4]. Differences in design and approach are discussed, as are implications for future work.

Introduction

Leigh Tesfatsion defines the field of *agent-based computational economics* (ACE) to be "the computational study of economies modeled as evolving systems of autonomous interacting agents" [6, p. 1]. Although research in the field has taken a wide variety of forms and directions [6], in general ACE is driven by the idea that economies are examples of "complex systems." That is to say, an economy is the sum of the interconnections and relationships between its parts, and thus the properties of the economy as a whole may be different than the aggregate properties of the parts [4, p. 2]. In many ways, such a system lends itself far more readily to simulation, which can incorporate its complexity, than to description by elegant equations, which must often drastically simplify their subject. Unlike traditional economic outlooks and techniques, ACE uses a bottom-up approach to economic modeling which allows for an arbitrary degree of specificity, heterogeneity, and complexity in the model. However, the very immensity and complexity of real economic systems that makes ACE so desirable also makes the construction of useful ACE models a daunting task.

Thus, the need arises for what Tesfatsion calls a *computational laboratory*: a framework which streamlines the construction of ACE models and which reduces the need for programming expertise on the part of the modeler [6, p. 17]. In spring 2009, we created EOS (Economics via Object-oriented Simulation), a computational laboratory which leverages the object-oriented nature of Java™ to describe the economic interactions and behaviors of markets and agents. In an attempt to improve upon its predecessor Minsim [2], EOS was designed with the goals of extensibility, refinability, and ease of use [1, p. 9] [5] [7]. The EOS framework uses three primitive types to describe ACE models, as well as an Economy object which acts as a simulation driver. Agents are decision-making entities, Goods represent commodities which Agents can own and use, and Markets provide a means for Agents to trade their Goods. The EOS documentation [1] provides a full description of the framework.

In addition to creating the framework, we implemented a “baseline” economy with three EOS goods: money, food, and labor. Labor is allocated to laborer agents on every time step and may be sold to farm agents, which use it to produce food. Farms then sell their product to the laborers, who must consume a sufficient quantity of food to in order to survive. In the baseline economy, the money supply is fixed, and transactions are resolved by a call auction mechanism, which maximizes trade volume and total surplus [3, p. 149]. The baseline is more thoroughly described in Rucinski's [5, p. 13] and Wang's [7, p. 8] papers.

After designing the framework and baseline, what remained was to demonstrate that EOS could be used to create a stable, yet justifiable simulation. In this paper, we use the term "stable" to describe two (often closely linked) features of a simulated economy:

1. *Population Stability* – The number of Agents in the economy should remain constant, or should converge to a level that is reasonable given the production capabilities defined by the model.
2. *Price Stability* – Although prices can (and probably should) fluctuate in a stable economy, they should not rapidly decay to zero or rise steadily to infinity. Inflation and deflation are features of real economies, but a simulation where these forces are consistently so extreme as to dominate all other price fluctuations would be of little use to experimenters.

By "justifiable," we mean simply that stability is achieved through means that are consistent with general economic theory or with common sense. It is often possible to force a simulation to grant the desired results through ad-hoc means, but such solutions both undermine the validity of an ACE model and substantially reduce its usefulness to later experimenters.

As Chris Rucinski [5] noted, designing agent behavior algorithms that lead to stability at the economy level is a challenging task. Agents must be able to adapt to changing prices and situations, yet must not act in ways which lead to the destabilization of the system. Furthermore, their actions must make sense on an individual level: the system should not require that agents consistently act for the good of anyone but themselves (at least in the simplest models). In our spring 2009 work on EOS, we managed to develop a pair of laborer implementations that achieved some of the desired characteristics. However, both implementations also exhibited some fundamental problems which motivated the work culminating in this paper. The first, the so-called BudgetLaborer (described by Rucinski [5]), could only maintain population stability when the population was far below the economy's theoretical carrying capacity (as determined by the farm's production function). For example, when the farm's food production as a function of labor was set to $F(L) = MAX\left(\frac{5}{2}L, 60\right)$, the simulation became unstable with more than five laborers per farm. Since each laborer consumed only a single unit of food each time step, and the farm could produce 60 units of food each time step without suffering from diminishing returns, a single farm should have been able to support up to 60 laborers. The SimpleLaborer alleviated this problem to some degree (24 SimpleLaborers could survive with a single farm), but suffered the additional problem of being unable to scale. That is, a simulation that remained population stable with a population of one farm and 24 SimpleLaborers would become unstable with two farms and 48 SimpleLaborers [A, fig. 1].

Thus, the EOS framework still lacked a truly functional baseline. To that end, we determined to develop a stable and justifiable implementation of a simple economy using the EOS framework. This involved designing a new I/O package to permit the running of experiments and analysis of data necessary to understand the problems with the spring 2009 work, the use of fixed-price money flow analysis to determine the existence of a stable baseline economy and to understand the dynamics of that economy, and the implementation of two more complex models, baseline 2 and baseline 3, to address issues unearthed by our analysis.

I/O in EOS

In many ways, a simulation tool is only as useful as its input and output capabilities. As of spring 2009, I/O in EOS was limited to a set of methods in a subclass of Economy which could be used to specify agents and markets to be added to the economy (input) or enable the gathering and printing of various types of data during the simulation run. The design of this I/O system, in which I/O was highly integrated into simulation code, placed severe limitations on the usefulness of EOS. Writing code to extract new types of data from the economy required modifying a subclass of Economy, and could not easily be reused across different Economy subclasses. Furthermore, the code for generically instantiating subclasses of Agent (which is complex due to the constructor arguments) cluttered the Economy subclasses and made implementation of new economies difficult. Thus, we decided that the analysis we would need in order to accomplish our goals both merited and provided an opportunity for redesigning I/O in EOS. To this end, we created a new I/O package, eos.io, containing 18 java classes. The tools it provides are described briefly here.

Input

The ability to rapidly code and modify new simulation classes [1, p. 5], which specify the parameters for a particular run of an EOS simulation, is vital to productivity when working with EOS. To support this, we created the BuildableEconomy interface, which contains several methods for initializing an Economy object. Subclasses of Economy can optionally implement this interface. More important is the Simulation class. Where an Economy object represents a single run of an EOS simulation, the Simulation class allows users to specify parameters and then generate multiple instances of Economy objects initialized with those parameters. Thus, a Simulation object truly represents an instance of a simulation. Simulations manage the seeding of random number generators to allow for repeatable results as well as the use of Printers (see

Output). They also create a log file to record the steps taken by the Simulation object in setting up a particular economy, which can be useful for debugging and keeping track of multiple simulation runs.

Output

The new output framework for EOS is based around three basic interfaces/classes. DataOutputEconomy is, like BuildableEconomy, an optional interface which Economy subclasses can implement to be compatible with the output capabilities provided by eos.io. DataPrinter objects extract specific data from a DataOutputEconomy, such as the price of food or the average wealth of laborers. Finally, Printer objects take the data extracted by their DataPrinters and output it in a specific format. The Printer objects which we implemented, for example, write output to Comma Separated Value (.csv) files. This system makes it easy to extract new types of data, since doing so simply requires writing the small amount of code needed to implement the DataPrinter interface. Printer objects, on the other hand, need only concern themselves with the format of the output. This enables the creation of Printers that support more complex and useful output formats. For example, the CSVMultiRunPrinter class allows the output data from multiple Economy objects to be concatenated into a single .csv file with a set of columns for each simulation run. Printer objects that produce graphical or command-line output could also be created. These tools proved vital to understanding and debugging our current work, and will hopefully be useful to EOS modelers and developers in the future.

Problems with the Baseline

With the new I/O framework in place, it became possible to understand more about the failings of the baseline farm and laborer implementations. Numerous experiments suggested a variety of possible culprits. For example, mass laborer deaths were often preceded by price spikes [A, fig. 2] and/or highly irregular trading volumes [A, fig. 3]. Another common event was a large scale shift in monetary wealth from laborers to farms or vice versa. Both circumstances could lead to stability issues. If most of the economy's fixed money supply ended up in the hands of laborers, then the possibility arose of food production grinding to a halt. If this was not corrected quickly, starvation would result. However, accumulation of money by the farm proved to be an even worse problem. Since baseline farms have no monetary outlet other than buying labor, a farm that could consistently make a profit would eventually end up with most of the money in the economy [A, fig. 4]. For obvious reasons, this brought a halt to economic activity. Furthermore, it created a justification problem, since the stability of the model essentially relied on firms being unprofitable.

Other parts of the baseline were difficult to justify as well. For example, the implementation of a baseline farm holds a sale when its food stock is greater than some target (this helped alleviate the money accumulation problem somewhat). This allows laborers to buy at low prices by effectively holding out and waiting for the farm to overproduce and lower its prices. However, forcing farms to have sales was difficult to justify since farms could have instead followed the same "holding out" strategy as the laborers. Indeed, the farm could simply wait for the laborers to become desperately hungry. Furthermore, although farms in the baseline could theoretically go bankrupt, since they only spent money to purchase labor (which always produced at least some food), the farms could never run out of both food and money in practice. This made justifying sales still more difficult, since the farm ran no risk by holding out for a better price. On the laborer side, stability required that laborers with sufficient food and money demand more for their labor, rather than competitively undercutting their less-fortunate comrades

[5, p. 25]. This, too, created a justification problem since the baseline provided no motivation on the level of the individual to support either predatory or cooperative labor pricing.

Fixed-Price Analysis: The Baseline

First, we tackled the problem of money accumulation. We began by examining the money flow, which can be visualized as a simple cycle [fig. 1]. Thus, if we know the prices of food and labor, it is very simple to calculate the farms' or laborers' net monetary earnings. For the purposes of this paper, we use the term *profit* to describe an agent's (or class of agent's) *net earnings of a particular good*. In a baseline economy with a single farm, the farm's monetary profit P_{money} for a given time step is simply $p_{food}V_{food} - p_{labor}V_{labor}$, where p is the price and V is the trade volume. In an economy with multiple farms, this equation gives the total monetary profit of all farms; each individual farm earns some fraction of P_{money}. For the purposes of this analysis, we will assume the single farm case. If the farm is not to accumulate any money, P_{money} must be zero. At carrying capacity, V_{food} must equal the number of laborers, since laborers require one unit of food each time step to survive. Finally, V_{labor} must equal the amount of labor required to produce V_{food}. Using the baseline's food production function $F(L) = MAX\left(\frac{5}{2}L, 60\right)$, F(L) is 60 (the theoretical carrying capacity) when V_{labor} is 24. Thus, for the farm to make zero monetary profit, $\frac{p_{labor}}{p_{food}} = \frac{5}{2}$, which is equal to the *productivity of labor when producing food.* We

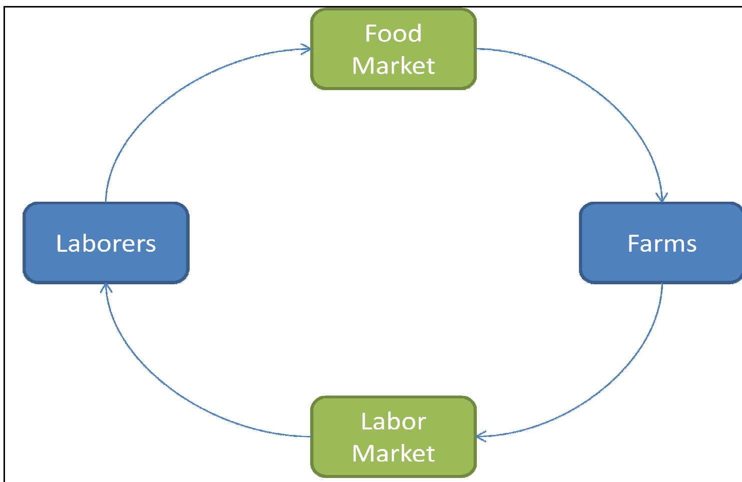

Figure 1: Baseline Money Flow

call this value the *stable price ratio* for the baseline economy. When the food and labor prices are related by this ratio, the baseline economy should be able to run stably to the extent that sufficient food will be produced without money accumulation occurring.

To affirm the economy's stability under these conditions, we created two new agent implementations: FixedPriceLaborer (FPL) and FixedPriceFarm (FPF), which as their names suggest buy/sell food and labor at fixed, agreed-upon prices that can be set to conform to the stable price ratio. Also important is the fact that the FPF's labor-selling algorithm only sells the amount of labor necessary to recoup monetary losses due to food purchases. Since the price of labor is fixed, this quantity is easily calculated. As predicted, the fixed-price bidding behavior allowed the simulation to run stably at carrying capacity with one farm and 60 laborers. Furthermore, it remained stable when scaled up linearly (e.g. five farms and 300 laborers).

We also performed experiments with the FPL and FPF economy where the ratio of the fixed prices was set to a value less than the stable price ratio. In these cases, there was initially a net flow of money to the farm. However, this caused laborers to die until V_{food} and V_{labor} (which are functions of the number of laborers) fell to a level where the farms' monetary profits once again dropped to zero [A, fig. 5]. This was an important result because it showed not only that a stable baseline economy can exist, but that an unstable one can become stable by a convergence process (in this case, the population converged to a stable level). In the reverse experiment, the fixed price ratio was set above $\frac{5}{2}$. In this case, the laborers accumulated money until the farms funds fell low enough to constrain V_{labor}. This caused the economy to reach a new V_{labor} equilibrium where the laborers made zero monetary profits [A, fig. 6]. Since a lower V_{labor} led to a correspondingly lower V_{food}, the laborer population also converged to a new equilibrium value.

These results tied the money flow problem to the justifiability problems. In the baseline economy, no agent could make a long-term monetary profit since this meant that other agents in the system were becoming bankrupt. Although analysis of the fixed price economy showed that such instabilities could be corrected by population convergence (death), in all cases such corrections had the effect of halting further monetary profits. In such a system, it seems, the only reasons for a farm to remain in operation are benevolent ones. Also, laborers choosing only to work enough to earn back the money they spend buying food is not justifiable in the baseline economy, since it neither earns the laborer more food nor more money. Instead, in the baseline this decision not to sell excess labor can only be seen as a benevolent action that gives laborers who need money a better chance of getting a job while also preventing the farm from overproducing (since the FPF always tries to achieve maximum production, only a shortage of labor or its own money can prevent it from doing so).

Baseline 2

To provide justification for these actions, we added a new good to the baseline economic model: leisure. Unlike food and labor, leisure is not a commodity which is bought and sold. Instead, it is awarded to laborers each time step in proportion to the labor which they do not or cannot sell. This change justifies the FPL's choice not to work when he has attained a certain stock of food/money, but does not allow farms to earn monetary profits since farms cannot be awarded leisure. To address this issue, we added firm ownership to the model via the interfaces SingleOwnerFirm (implemented by farm classes) and SingleOwnerFirmOwner (implemented by laborer classes). Among other things, firm owners can withdraw money and other goods from the firms they own. Thus, if an owner's firm is sufficiently profitable, the owner need not work and can instead convert his or her entire labor allowance into leisure on each time step. We named this new model baseline 2, and added the class BL2LazyOwner (LO) to represent a firm owner who will always choose to request a salary from his or her firm(s) rather than work. Like the FPL and FPF implementations, the initial LO implementation makes bids for food at a fixed price.

Fixed-Price Analysis: Baseline 2

We then proceeded to perform a similar fixed-price analysis of baseline 2. The changes made to create baseline 2 still do not allow the economy to run stably while any agent consistently accumulates money. When the money supply is fixed, such money accumulation will be a problem in any model. Instead, the model must provide mechanisms (consumption or investment options, for example) which allow any money accumulated by successful agents to circulate back into the economy. In baseline 2, leisure effectively creates a consumption outlet by motivating wealthy owners and laborers not to work when they have extra money available. The extra money then gets spent on food [fig. 2]. This means that we can once again use money flow to derive the stable fixed-price ratio by setting P_{money} to zero for all agent classes.

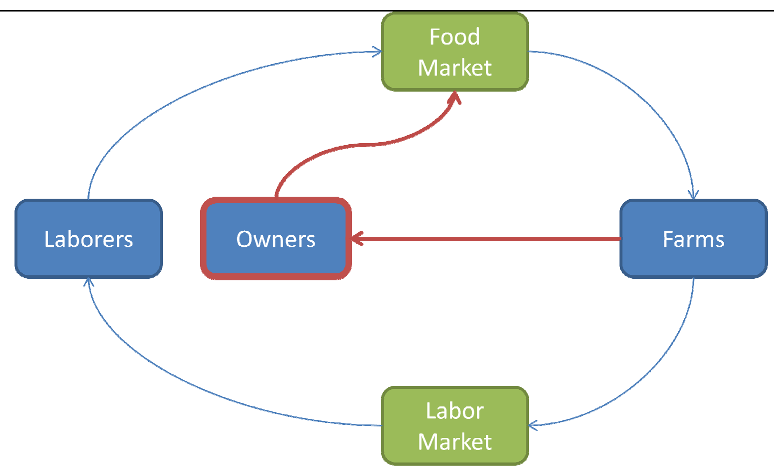

Figure 2: Baseline 2 Money Flow

Changes from the baseline money flow diagram are highlighted in red.

In baseline 2, the farm's income is once again $p_{food}V_{food}$. However, in addition to spending $p_{labor}V_{labor}$, it also must pay a salary S to its owner. If the owner is not to work and not to accumulate any money, then S must be p_{food}, which is the owner's only expense.

Thus, the farm's monetary profit $P_m = 0 = p_{food}V_{food} - p_{labor}V_{labor} - p_{food}$. This sets the stable price ratio $\frac{p_{labor}}{p_{food}}$ at $\frac{V_{food-1}}{V_{labor}}$. Using the same production function as in the baseline, at carrying capacity one farm should be able to support 60 laborers (59 FPLs and one LO), leading to a price ratio of $\frac{59}{24}$. At this price ratio, simulations of the baseline 2 economy remained stable over many time steps. Furthermore, unlike in the baseline, these prices allowed both farms and laborers to "profit" from their economic activity. Laborers earned enough money to afford them some time for leisure, while farms earned enough money to allow their owners to survive without working (thus the owners made a profit in leisure) [A, fig. 7]. Baseline 2 thus represented a significant step towards solving the money flow and justifiability problems of the baseline. What remained was to design agent bidding algorithms which would allow the economy to be as stable and nearly as efficient when allowing prices to fluctuate as in the somewhat trivial fixed-price case.

Dynamic Pricing: Baseline 2

Designing such bidding algorithms is a complex process not only because the decision-making involved is complex but also because the algorithms must conform to several criteria in order to function properly and be worthy of consideration:

1. *Agents on both the supply and demand sides must make dynamically-priced bids*. If, for example, laborers are willing to pay more for food when they are starving but farms never raise prices, the price will not change (assuming that we are using a call auction).
2. *Agents must implement mechanisms that can drive the price both up and down*. For example, it might seem reasonable that laborers normally use the last market price for their food bids but are willing to pay ten percent extra when their food stocks are

especially low. However, this bidding algorithm would lead to a steadily rising food price since no laborers ever make bids that are below the last market price.

3. *Bidding algorithms must provide some way to quickly escape a price "stand-off."* If the highest buy bid is lower than the lowest sell bid on a particular time step, the trade volume will be zero and the market price will not change. In such cases, agents must continue to adjust their bids until a mutually agreeable price is once again found.
4. *Bidding behaviors must realistically model agents' goals*. Like other aspects of the model, it is important that bidding algorithms produce justifiable behavior in the sense that they are inspired either by economic principles or by ways in which people make economic decisions in the real world.

To begin, we designed agent bidding algorithms for food. Several methodologies were tried before an algorithm was found that allowed the economy to run in a stable manner. One of the failed strategies is worth mentioning here because it will inspire a methodology used later in this paper. Price-seeking bidding is a strategy where the agent assumes that he or she is a price taker, and thus that his or her goal is to seek out the "true" price of a good. An agent trying to buy food, for example, would follow this procedure: if the agent succeeds in buying food at a price p, then make the agent's next bid at $p - \Delta$, since perhaps the agent overpaid. If the agent failed to buy food at p, however, then make the agent's next bid at $p + \Delta$ in order to be more competitive. This algorithm was fairly successful at getting the buy- and sell-side bids to converge to some price. However, since it only takes into account the price of one good at a time, it does not assist convergence to a price that is consistent with the derived stable price ratio.

The algorithm that eventually proved successful used a supply/demand curve strategy inspired by basic microeconomic theory. Agents using this algorithm create a logical supply or demand curve based upon several factors such as current prices and the agents' current supplies. The agents then make several bids along their calculated curves. For example, the laborer's food demand curve starts at (zero quantity, labor price) and slopes downwards to the point (desired food stock, food price). Then for each unit of food between the laborer's current food stock and 110% of the desired food stock, the laborer makes a buy bid using the demand curve to determine the price for that bid [A, fig. 8].

Using the demand and supply curve bidding algorithms, we were able to run stable baseline 2 simulations where the food prices were allowed to fluctuate. In such simulations, it was unnecessary to fix initial prices to match the stable price ratio. Indeed, when the initial food price was set to an unstable value with respect to the fixed labor price, the agent's bidding behavior caused the food price to shift fairly rapidly to the stable value [A, fig. 9]. Furthermore, these simulations could be run at a relatively high population to theoretical carrying capacity ratio (> 93%). This represented a huge improvement over the baseline implementations, where the highest stable population to carrying capacity ratio achieved was 40%.

Problems with Baseline 2

However, when we implemented the supply and demand curve bidding algorithms for labor and food (thus allowing both prices to fluctuate), two significant problems arose. The first was an issue of productivity that arose from fundamental differences between dynamically pricing food and dynamically pricing labor. When buying and selling food with dynamic prices, wildly fluctuating trade volume does not cause instability because food can be stored. If a farm overprices its food and thus sells very little of it on a particular time step, it can simply attempt to sell the same food again on a later time step for a more reasonable price. On the other hand, if a laborer fails to sell his or her labor or a farm fails to purchase the labor it needs, the opportunity to do so is lost forever. Furthermore, due to the diminishing returns of the food production function, a farm cannot simply “make up” for lost production on previous time steps by buying a larger quantity of labor unless it is producing far below capacity. When wages were allowed to fluctuate, this effect greatly lowered the productivity of the economy, and thus reduced the number of laborers that a given farm could support.

Another problem was that of price drift. Since both prices were free to move, they tended to drift in one direction or the other rather than oscillate around a relatively stable value [A, fig. 10]. Conceptually, the reason for this is fairly simple. Since the stable price ratio can be maintained at arbitrarily high or low prices, it does not act to pull the prices towards any particular point. Thus, when a price changes (due to a shortage, for example), it creates a dual response that effects both prices. For example, consider the response to a rise in the price of food. This will cause farms to buy a larger quantity of labor and produce more until the demand is met, allowing the food price to once again fall. However, a higher food price will also lead farms to pay more for the labor they buy, thus driving up the price of labor. Even if the price ratio stays close to the stable price ratio, the consequence of this response and the analogous responses to other possible price perturbations is that a change in one price will pull the other price in the same direction.

Of course, if “random” surpluses and shortages occur with equal probability, this response should still lead to oscillation rather than drift. However, the aforementioned productivity problem prevented this from being the case. When the population is small (< 75% of the theoretical carrying capacity), farms tend to overproduce, so laborers are consistently able to maintain large stocks of food. In these cases, prices fall steadily as in [A, fig. 10]. When the population is near carrying capacity, farm productivity is insufficient to support the population.

Desperate laborers drive prices up, leading to steadily rising prices until starvation claims enough agents to switch the economy over to the overproduction (falling prices) case [A, fig. 11]. At borderline population values (~75% of carrying capacity), both rising and falling prices can be observed, although the falling trend dominates [A, fig. 12].

To try to curtail both the productivity loss and price drift effects of allowing the labor price to fluctuate, we attempted to increase the "stickiness" of prices by modifying agent bidding algorithms. For example, rather than using the last market prices when constructing supply and demand curves, we used exponentially smoothed estimates so that agents' perceived value of goods would reflect historical as well as current prices. However, this only slowed, rather than halted, the problems described above [A, fig. 13].

Thus, while baseline 2 solved some of the justifiability and money flow problems of the original baseline, it suffered from new problems when both food and labor prices were allowed to fluctuate. Furthermore, baseline 2 was still limited by a stable price ratio. Although the addition of firm owners provided a consumptive outlet for profit-making farms, the amount of money transferred in this way was both small and relatively invariant since the owners took only as much as was necessary to buy food each time step. This need for an exact price ratio in order to maintain stability surely added to the challenge of creating effective bidding algorithms.

Baseline 3

Using our experiences with baseline 2, we created a final baseline-tier model: baseline 3. Where in baseline 2 owners and laborers struggled to earn leisure by not working, in baseline 3 they strive to acquire a good called *utility*, which represents happiness in a general sense. Like leisure, unsold labor is converted to utility at the end of each time step. However, utility can also be produced and sold by utility factories, which hire labor to produce their product. In our baseline 3 implementations, the price of utility is fixed, while the prices of food and labor are allowed to move freely.

At first, the production and sale of an abstract good like utility may seem unrealistic and difficult to justify. However, in baseline 3, utility can be thought of as representing different things depending on its source. For example, utility earned by not working represents the pleasure of leisure, while utility produced by firms represents consumable goods like candy, gourmet foods, or entertainment. Regardless of its source, utility of any sort contributes to a laborer or owner's overall contentment. Thus, the utility good in baseline 3 is essentially the simplest possible model of preferences and enjoyment of consumer goods. What is more difficult to justify is fixing of the utility price. Why should the utility price be fixed when the food and labor prices are allowed to fluctuate? One possible justification is that, since utility represents a large number of different entertainment-type goods, each of which is produced and sold in small volumes, the price of utility would be far stickier than that of food or labor. As more different goods are added to an EOS model, prices will naturally become stickier since it becomes more difficult for prices to move together. Thus, fixing the price of utility can be seen as an attempt to model this phenomenon while avoiding the complexity of implementing many different utility goods.

A second smaller, but still relevant new feature of baseline 3 is its use of more complex production functions. As previously stated, baseline 2 uses a capped linear production function: $F(L) = MAX\left(\frac{5}{2}L, 60\right)$. While such a function provides some realism by limiting the production capacity of any one firm, it fails to reflect the steadily diminishing returns (a. k. a. diminishing marginal product) characteristic of many real production functions [3, p. 273]. Instead, baseline 3 uses quadratic production functions, which provide diminishing returns in the form of a linear, downward-sloping marginal product of labor (*MPL* is the derivative of production) curve [A, fig. 14]. The utility and food production functions arbitrarily differ by a factor of two. In addition to being more justifiable than the production functions in baseline 2, the quadratic functions in baseline 3 suggest a simple, yet intelligent bidding algorithm for the purchase of labor by firms. If a firm assumes the price of its product to be p_p, then it can calculate the marginal revenue that would result from hiring the i^{th} laborer (MR_i) using $MR_i = p_p * MPL_i$. The firm is thus willing to pay up to MR_i to hire the i^{th} laborer. Both the farms and factories in baseline 3 use this bidding strategy.

Money Flow Analysis

The addition of a utility market in baseline 3 changes the money flow in significant ways [fig. 3].

Where a firm owner's "consumption" in baseline 2 was limited to not working (and thus receiving a salary equal to the price of food), in baseline 3 the utility market acts as a flexible, relatively unbounded outlet for the accumulated wealth of successful owners and

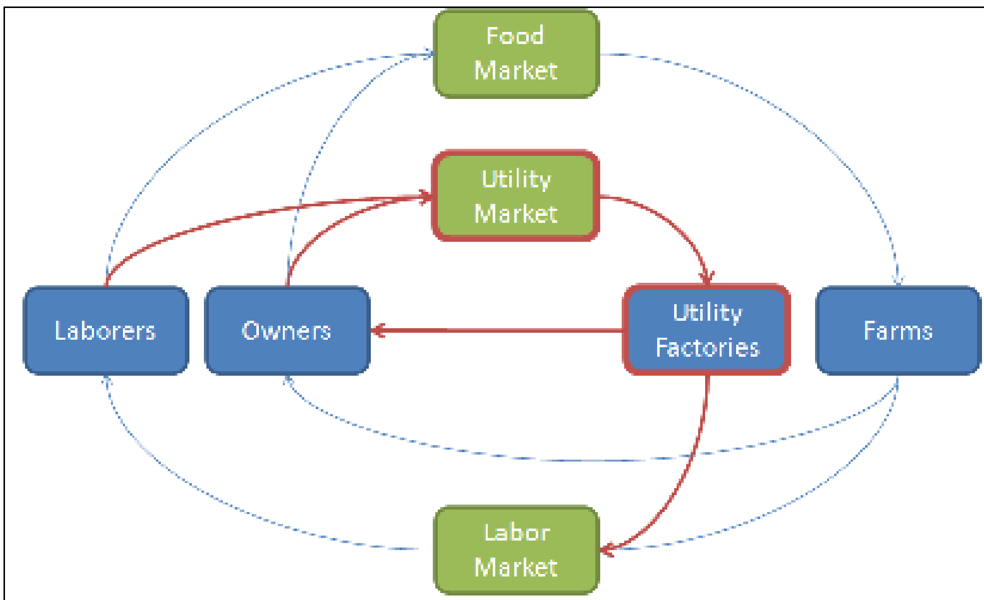

Figure 3: Baseline 3 Money Flow

Changes from the baseline 2 money flow diagram are highlighted in red.

laborers. This in turn removes the problem of high firm profits leading to money accumulation and therefore instability. In baseline 3, owners take and spend all of their firms' profits, effectively preventing monetary accumulation. Furthermore, since acquiring utility is the end goal of owners in baseline 3, their actions are easily justifiable.

The effect of this unbounded consumptive outlet is to remove the stable price ratio constraint that characterized the money flow of the baseline and baseline 2 models. Where before we could derive such a ratio by setting the farm's P_{money} to zero (along with the P_{money} of every other agent), placing the same constraint on P_{money} in baseline 3 does not constrain the prices. Consider the following monetary profit equations based on Figure 3 and the above description of owner behavior:

Agent Type	**Money IN (net to all agents of this type)**	**Money OUT (net from all agents of this type)**	**Notes**
Farm	$V_{food}p_{food}$	$v^F_{labor}p_{labor} + (V_{food}p_{food} - v^F_{labor}p_{labor}) = V_{food}p_{food}$	v^F_{labor} is the volume of labor purchased by farms. The $(V_{food}p_{food} - v^F_{labor}p_{labor})$ term comes from the fact that the farm's owner withdraws all monetary profits from the farm.
Factory	$V_{util}p_{util}$	$v^U_{labor}p_{labor} + (V_{util}p_{util} - v^U_{labor}p_{labor}) = V_{util}p_{util}$	v^U_{labor} is the volume of labor purchased by factories. The equations are essentially equivalent to those of the farm.
Laborer	$V_{labor}p_{labor}$	$Lp_{food} + v^L_{util}p_{util}$	L is the number of laborers, and v^L_{util} is the volume of utility purchased by laborers.

Owner	$(V_{food}p_{food} - v_{labor}^{F}p_{labor}) + (V_{util}p_{util} - v_{labor}^{U}p_{labor}) = V_{food}p_{food} + V_{util}p_{util} - V_{labor}p_{labor}$	$Wp_{food} + v_{util}^{W}p_{util}$	W is the number of owners, and v_{util}^{W} is the volume of utility purchased by owners.

Table 1: Money Flow Expressions for Baseline 3

Setting the money in and money out equations equal to each other for an agent type in Table 1 is equivalent to setting P_{money} to zero for that agent type. It is easy to see that setting P_{money} to zero for farms and factories gives us a trivial result in baseline 3. This corresponds to the fact that owners withdraw all of their firms' profits as a salary.

Next, we consider the laborers, where setting P_{money} to zero gives us $V_{labor}p_{labor} = Lp_{food} + v_{util}^{L}p_{util}$. $v_{util}^{L} \geq 0$, so $V_{labor}p_{labor} \geq Lp_{food}$ and thus $\frac{p_{labor}}{p_{food}} \geq \frac{L}{V_{labor}}$. Thus, this equation only gives us a lower bound for the labor price to food price ratio. Even the more complex owner equation fails to give us a fixed ratio:

$$V_{food}p_{food} + V_{util}p_{util} - V_{labor}p_{labor} = Wp_{food} + v_{util}^{W}p_{util}$$

$$(L + W)p_{food} + (v_{util}^{L} + v_{util}^{W})p_{util} - V_{labor}p_{labor} = Wp_{food} + v_{util}^{W}p_{util}$$

$$V_{labor}p_{labor} = Lp_{food} + v_{util}^{L}p_{util}$$

Instead, it can be simplified to the laborer's P_{money} equation (Note that the expressions in Table 1 assume that P_{food} (food profit) is also zero for each agent type. That is to say, no agent attempts to consistently accumulate food. Instead, laborers and owners try to maintain a comfortable stock of food. This allows us to conclude that $V_{food} = L + W =$ *the laborer/owner population*. This lack of a fixed stable price ratio is important both because it leaves more room for imperfection in bidding algorithm design (which may allow for simpler algorithms that more closely resemble consumers' real-world decision-making) and because it means that baseline 3 economies could converge to or oscillate about different (relative) price equilibria. This creates room for interesting experiments to determine what factors cause an economy to settle on one particular price ratio and which agents benefit from or are disadvantaged by such a result. The results of a simple version of such an experiment are described below. Finally, it is important to note that the specific parameters of the production functions may impose additional limits on possible equilibria. For example, the utility production function, along with the number of utility factories in the economy, determines a maximum possible value of V_{util}. This in turn provides an upper bound on v_{util}^{L}, and thus impacts the range of possible price values. Examining the extent of this effect (both mathematically and in actual simulations) could be an important future step, as it would give insight into the importance of using highly realistic versus relatively arbitrary production functions in simulation.

Bidding Algorithms

The bidding algorithms used by the agents in our final baseline 3 implementations are designed around an idea of "complex buying and simple selling." When buying goods, agents in baseline 3 use demand curve algorithms to express how much they are willing to pay for particular goods. Firms buy labor using the *MPL/MR* algorithm described above, while laborers and owners buy food using the baseline 2 demand curve algorithm [A, fig. 14]. These algorithms are "complex" because they leverage information outside of the current market price of the particular good (such as the prices of other goods) in evaluating how much the agent is willing to bid. When selling goods, on the other hand, agents use a very simple random-seeking algorithm similar to the price-seeking algorithm described earlier. In an attempt to discover the "true" price of a good, agents bid at random values close to the market price. Over time, this has the effect of locating an efficient price.

Experiments

To further validate baseline 3, we performed several simple experiments. First, we verified that the stability exhibited by the economy was not a temporary phenomenon by running the simulation for 100000 time steps (with 200 agents). We found that the economy remained stable in both population and prices over this period [A, fig. 15]. In another basic stability test, we ran the simulation with a large population of 1950 agents (50 farms, 50 factories, 100 owners, and 1750 laborers). This economy also remained very stable throughout the simulation period [A, fig. 16]. We then performed a test to examine how often such stable runs were produced. In 100 runs of the simulation (10000 time steps, 117 agents), we found that 100% of the runs were population stable (none of the agents died).

A final experiment was motivated by the above money flow analysis, which suggested that baseline 3 economies could be stable at a variety of price equilibria. To test the extent to which our current laborer and firm implementations could reach different equilibrium price ratios, we ran 50 instances of the simulation, recording average prices and trade volumes as well as utility distribution (how much utility was accumulated by laborers versus owners). In each simulation instance, each agent's starting supply of goods was randomly adjusted to a value between 90% and 110% of its original value. A different random seed was also used on each run. The results of this experiment showed that small changes in the initial conditions, combined with randomness in the market clearing and bidding algorithms could lead the economy to reach slightly different equilibria. Furthermore, the resultant equilibrium affected the relative and absolute welfare of laborers and owners [A, fig. 17].

Discussion

Establishment of a Stable and Justifiable Baseline

The baseline 3 model and accompanying agent implementations can thus be used to simulate a stable and justifiable baseline economy in EOS. Using the new I/O package, the stability of baseline 3 simulations has been demonstrated over numerous runs and in special cases such as very long-running and very large simulations. Baseline 3 incorporates two major modifications to the original baseline model. First, baseline 3 contains a fourth good, utility. As previously discussed, the presence of utility significantly changes the economy's money flow. By permitting and providing justification for consumption and leisure, the utility good helps to keep the economy's fixed money supply circulating. It also provides a means for assessing the relative success of agent implementations beyond the ability to survive. Furthermore, baseline 3 provides a model for firm ownership. Not only does firm ownership increase realism, but also it is a crucial element of the model because it allows firms to operate in a profit-maximizing fashion without accumulating money since firm owners can withdraw and spend their firms' profits. Utility/leisure provides an outlet for excess money earned by "human" agents; ownership provides such an outlet for firms.

At the strategy level, the agent implementations created to inhabit baseline 3 utilize new bidding algorithms which are more effective and in some cases more justifiable than those used in the baseline (in particular, firms no longer have arbitrary sales). The labor-purchasing algorithm used by firms takes advantage of the quadratic production model used in baseline 3 to make particularly rational and justifiable bids, while the random-seek algorithms used for selling provide a simple mechanism for finding a valid price. The baseline 3 bidding algorithms allow both the price of food and the price of labor to fluctuate, producing interesting oscillations while avoiding extreme price drift.

A Comparison of Approaches

The purpose of this work was to examine and expand the usefulness and viability of the EOS framework as a computational laboratory by creating a stable and justifiable baseline model. Thus, it is useful to compare our efforts to a contemporary project which attempts a similar goal. In his 2008 thesis, Marc Oeffner attempts to create a "reasonably validated agent–based macroeconomic simulation model" [4, p. 3]. In many ways, Oeffner's work and approach differ substantially from our efforts with EOS. For example, Oeffner describes a complex validation and calibration procedure [4, p. 35]. Although he states that the validation process is "optimally conducted on multiple levels… down to the individual agent level" [4, p. 38], Oeffner's process validates the model only on the macro level by comparing the model's macroeconomic properties to those of a reference system [4, p. 48]. Oeffner sees his goal as developing his model (called Agent Island) to the point where it has "the ability to reproduce observed aggregate phenomena based on individual agents and their interactions" [4, p. 49]. He uses the concept of 'generative sufficiency' to assert that, since any micro-level agent model that produces valid macro-level output is a candidate for being the correct micro model, formal micro-level validation is unnecessary unless multiple such candidates are found [4, p. 49]. In such a case, further work can compare the candidates at a micro-level to determine which is more likely to be correct. In developing baseline 3, for EOS, on the other hand, we took great care to ensure that micro-level actions were justifiable, and limited macro-level validation to general requirements such as stability.

There are advantages and disadvantages to each approach. On one hand, Oeffner's calibration and validation procedure might provide and easy (and even automated) way to "fit" an economic model to a particular set of empirical data. On the other hand, such a process discourages the development of individually validated libraries of agent and market implementations. Such libraries would greatly improve the usability of a computational laboratory since they would allow experimenters to rapidly construct simulations tailored to their particular needs. Furthermore, our method seems more amenable to examining hypotheses in the form “if agents/markets behave like X, then the macro-level result will be Y.” This capability is important if our goal is to supplement purely mathematical analysis of such claims by simulating the economy as a complex system. Finally, our experience with the baseline showed that unjustifiable agent behavior can indeed produce seemingly acceptable macro-level results in a highly artificial manner. For example, the baseline farm/laborer implementations can produce stable simulations with oscillating prices much like those produced by baseline 3. However, the baseline economy is only stable under a narrow set of conditions and relies upon unjustifiable agent behavior to maintain its stability. Without micro-level validation, it seems difficult to claim that a particular model whose output is matches empirical data is correctly modeling the underlying dynamics of the system.

Another major point of difference between baseline 3 and Agent Island is the relative complexity of the two models. In general, Oeffner’s model is far more complicated than anything attempted thus far in EOS. Where baseline 3 models utility as a single good, Agent Island uses two utility-like goods (hash and beans) which appeal in different degrees to different agents based on agents’ particular utility functions [4, p. 93]. Where baseline 3 uses simple quadratic production functions, Agent Island uses sophisticated production functions that take into account the production technology of each firm [4, p. 110]. Oeffner’s Firms can even purchase better technology via a capital goods market. To further complicate things, the capital goods in Agent Island are non-homogenous (not perfect substitutes for each other), and the capital goods market is modeled as a “monopolistically competitive” market that functions differently from the hash and beans markets (which work much like baseline 3's call auction) [4, p. 126]. Yet another model is used to implement Agent Island's labor market. Where leisure is an important component of baseline 3, in Agent Island households (laborers) are obligated to work to supply the labor demand of firms. Oeffner justifies this oddity by the fact that his study focuses on inflation, and thus labor market issues such as unemployment can be ignored [4, p. 113]. Perhaps the greatest complexity difference, though, is Oeffner's implementation of financial instruments such as debt, savings, investment, credit markets, and a central bank. Where firms in baseline 3 have a single owner, for example, Agent Island firms are "owned by the workers in equal shares" [4, p. 103]. In baseline 3, agents who fail to purchase enough food starve to death; in Agent Island, they can borrow money to maintain a minimal level of consumption [4, p. 87].

This complexity in some ways makes Oeffner's model far closer to something useful for economic research than baseline 3, but it is also eerily reminiscent of EOS's predecessor, Minsim. Like Agent Island, Minsim's core economic model incorporated many complex elements such as banking, debt, and a specialized labor market [2]. Unfortunately, this complexity made Minsim extremely difficult to modify, debug or even understand. With EOS, we hope to develop our model incrementally. By analyzing and experimenting with each major change to our model, as has been done in the current work with the baseline, baseline 2, and baseline 3 models, we hope to work towards a model with the same level of complexity as Minsim or Agent Island, but whose components have been individually tested, studied, and verified. Another advantage of the EOS approach is that it makes our model more easily adaptable to a wide variety of experiments. Agent Island, on the other hand, is so specialized for the study of inflation that there may be little in the model that could be directly re-used in a different experiment.

Conclusion and Future Work

Examining the differences between our work with the EOS computational laboratory and Oeffner's development of Agent Island not only provides a fresh perspective on our current progress and approach but also helps to suggest directions for future work. Oeffner's work on implementing economic theories on household savings and other complex phenomena in an ACE model could, for example, be used to develop realistic implementations of such phenomena using the EOS framework. By building such implementations atop EOS primitives, we can incorporate the complexity of Oeffner's model into EOS without limiting EOS's flexibility. For example, if a Contract framework primitive is developed for EOS, it could be used as a basis for modeling debt, shared firm ownership (via stock), and other financial instruments [1, p. 2]. Oeffner's validation and calibration procedure could also be used to bring EOS models closer to modeling observed economic behavior. However, the benefits of this top-down calibration approach would have to be weighed against those of a bottom-up adaptation approach, such implementing machine learning algorithms to direct agent behavior.

Our construction and analysis of a stable baseline model using the EOS framework represents a significant step in the process of developing and verifying the EOS framework as an ACE computational laboratory. Nonetheless, there is still far more work to be done. Hopefully, the description and analysis presented here indicates the difficulty encountered and the degree of care required in conscientiously designing an agent-based economic model. However, since there are arguably few problems of such practical importance as understanding the complex workings of macroeconomics, ACE is certainly worth the effort.

References

1. Adelson, M. (2009). *EOS documentation* (Unpublished Documentation). Retrieved from http://eos.cs.princeton.edu/madelson_documentation05.16.09.pdf

2. Chan, C. (2008, May 5). *An agent-based model of a minimal economy* (Unpublished Paper). Retrieved from http://minsim.cs.princeton.edu/resources/ChrisChan_IW_2008_Part2.pdf

3. Mankiw, G. (2007). *Principles of microeconomics* (4th ed.). Mason, OH: South-Western Cengage Learning.

4. Oeffner, M. (2008). *Agent–based Keynesian macroeconomics — An evolutionary model embedded in an agent–based computer simulation* (Doctoral dissertation, University of W¨urzburg, W¨urzburg). Retrieved from http://www.opus-bayern.de/uni-wuerzburg/volltexte/2009/3927/pdf/OeffnerDissohneAnhang.pdf

5. Rucinski, C. (2009, May 4). *EOS: developing a general agent-based economic simulation* (Unpublished Junior Independent Work). Retrieved from http://eos.cs.princeton.edu/rucinski_spring_jp.pdf

6. Tesfatsion, L. (2002). Agent-based computational economics: growing economies from the bottom up. *Artificial Life, 8*(1), 55-82.

7. Wang, Y. (2009, May 4). *Agent-based computational economics: building beyond Minsim* (Unpublished Junior Independent Work). Retrieved from http://eos.cs.princeton.edu/codywang_JP_writeup_eos.doc

Appendix

1.

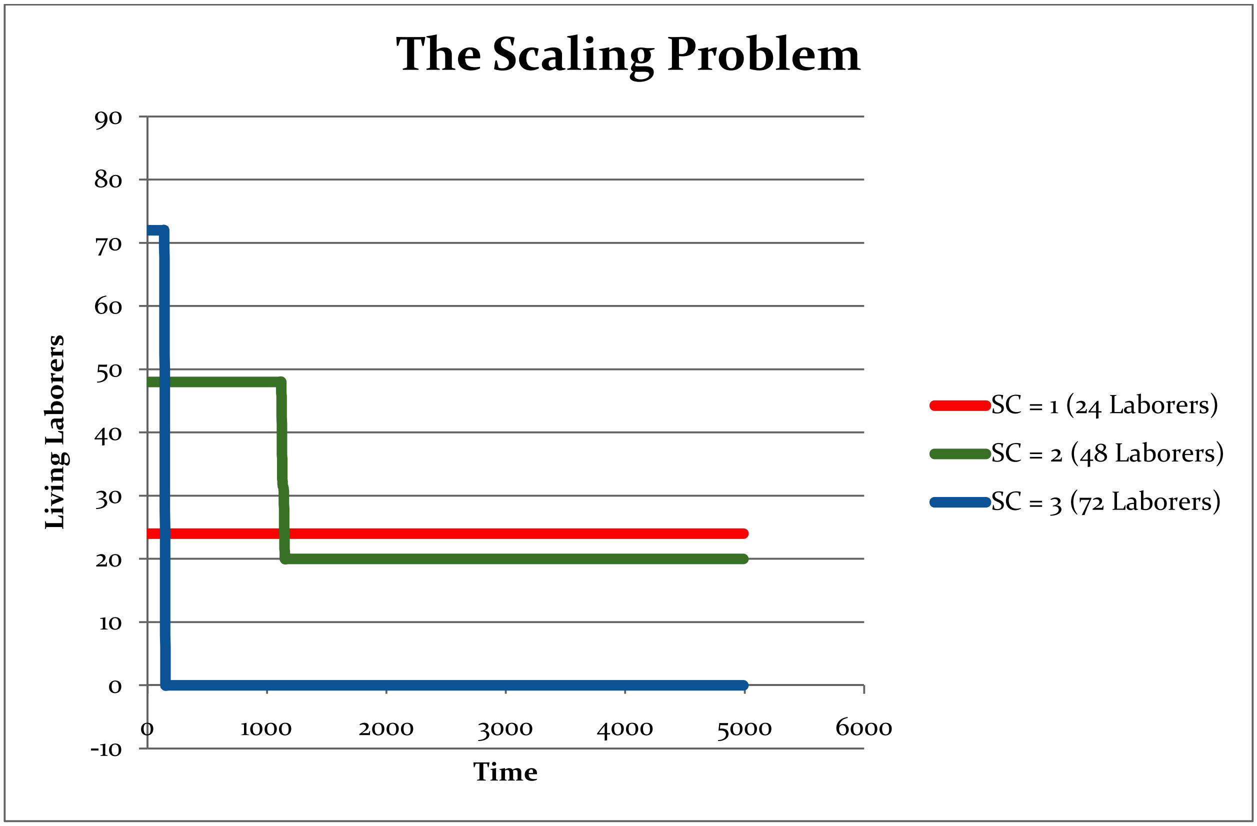

Figure 1

This plot shows the loss of stability as a baseline economy with a 24:1 laborer to farm ratio was scaled up in size by a factor *SC*. Rather than scaling linearly, increasing the economy's population caused many or even all of the laborers to starve.

2.

Figure 2

This plot illustrates an example simulation run where dramatic price spikes preceded mass laborer deaths (the blue dashed line).

3.

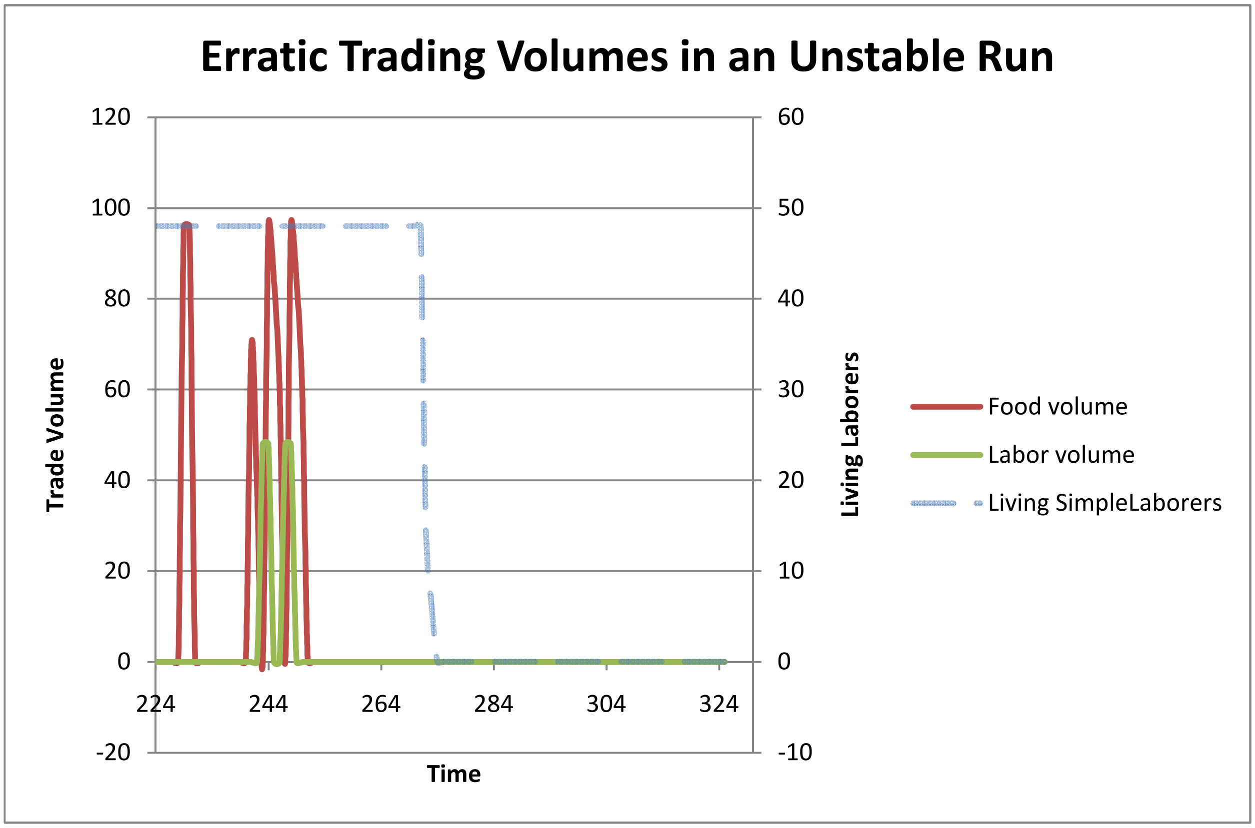

Figure 3

This plot illustrates an example simulation run where highly erratic trade volumes preceded mass laborer deaths (the blue dashed line). This suggests that the farms and laborers were unable to "agree" upon any price for very long.

4.

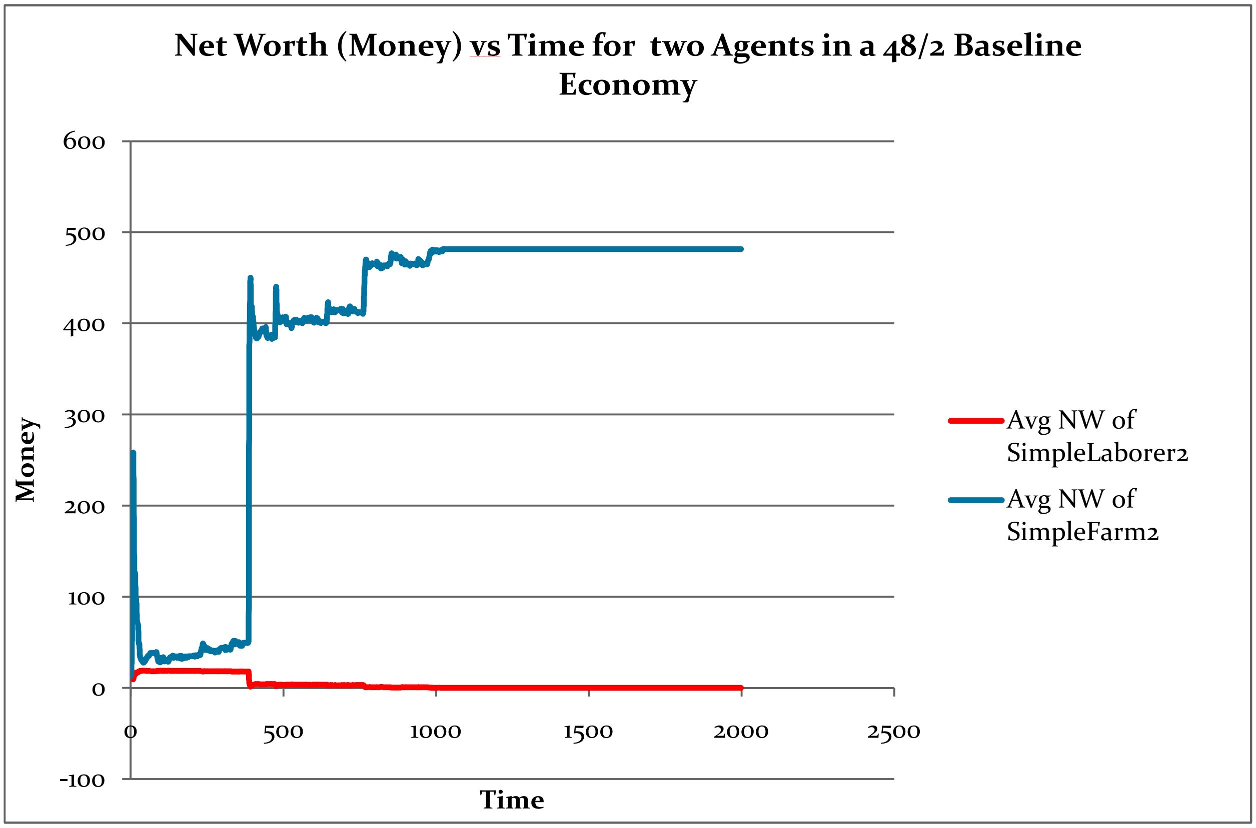

Figure 4

This plot tracks the net worth (money) of two agents in a baseline economy with 48 laborers and 2 farms. All of the laborers in this simulation run died shortly after the farm reached its peak net worth.

5.

The following plots (5A-C) show the results of a fixed-price baseline simulation run where the fixed-price ratio was set to two instead of the derived stable price ratio of $\frac{5}{2}$.

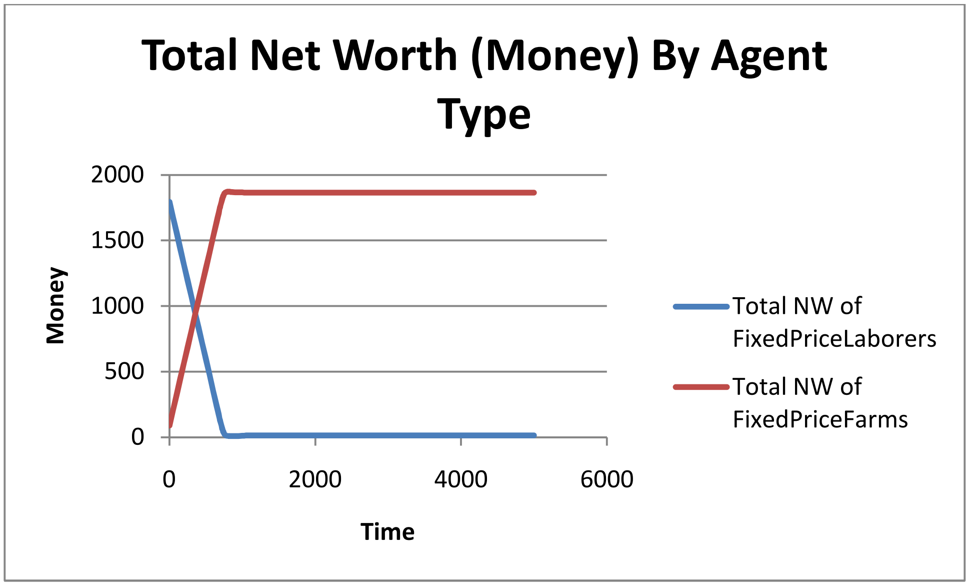

Figure 5A

Since the fixed-price ratio is smaller than the stable price ratio, the farm was able to accumulate money. Eventually, it acquired nearly all of the money supply.

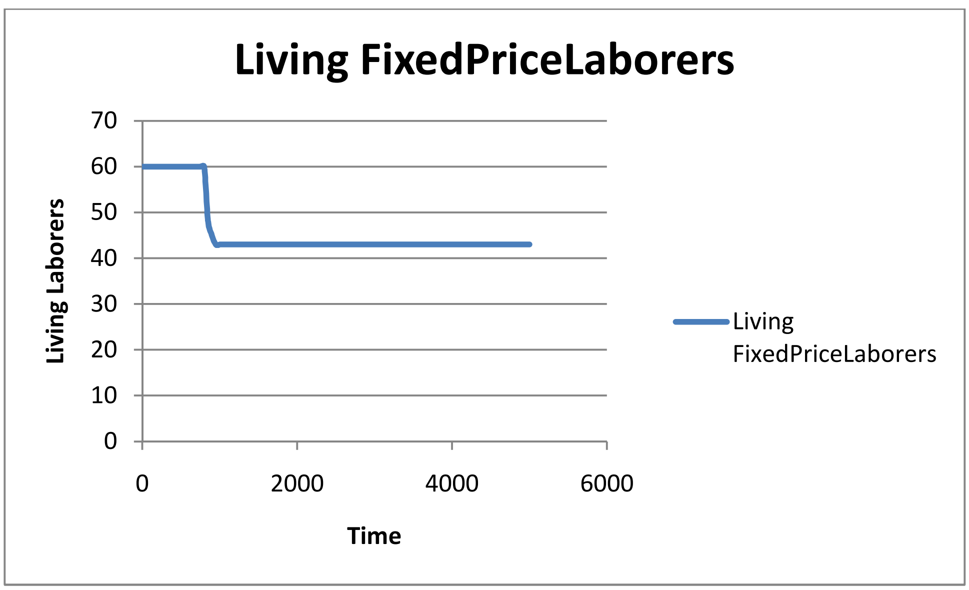

Figure 5B

Under these conditions, some, but not all of the laborers ran out of money and starved to death.

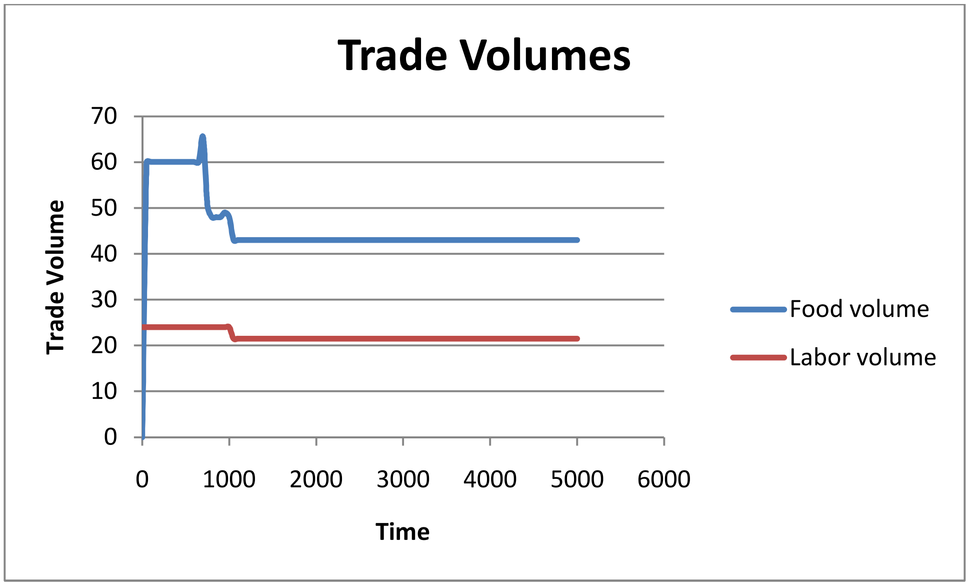

Figure 5C

As the population fell, so too did the trade volumes for food and labor. Eventually, the volumes reached new equilibrium values of 43 food and 21.5 labor. At a fixed-price ratio of two, the farm's monetary profit was reduced to $43 * p_{food} - 21.5 * 2 * p_{food} = 0$. Thus, the economy restabilized at this lower population. However, since monetary profits were once again zero across the board, the surviving laborers were unable to reclaim any of the money they had lost to the farm.

6.

These plots (6A-C) show the results of a fixed-price baseline simulation run where the fixed-price ratio was set to 2.85 instead of the derived stable price ratio of $\frac{5}{2}$.

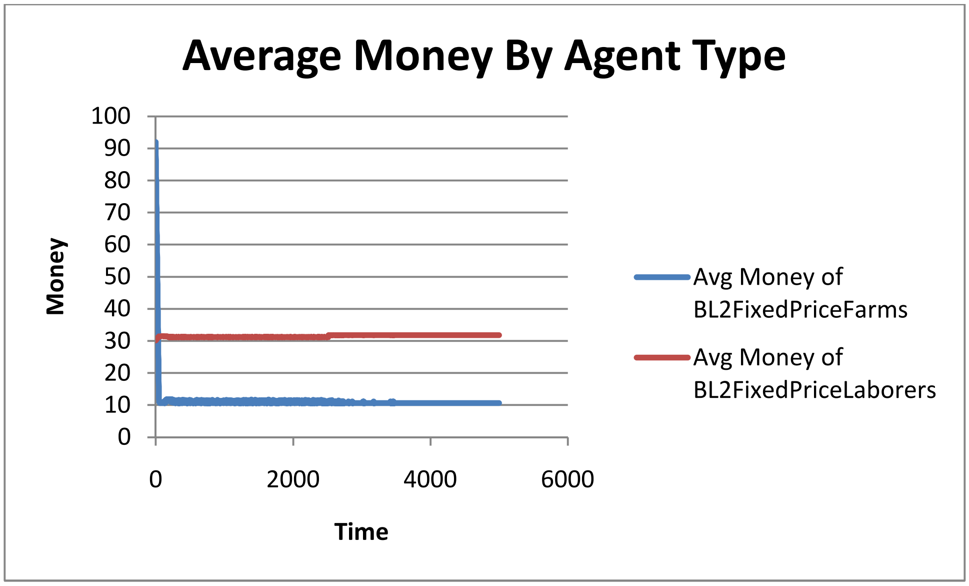

Figure 6A

Since the fixed-price ratio was set higher than the stable price ratio, the farm lost most of its money to the laborers.

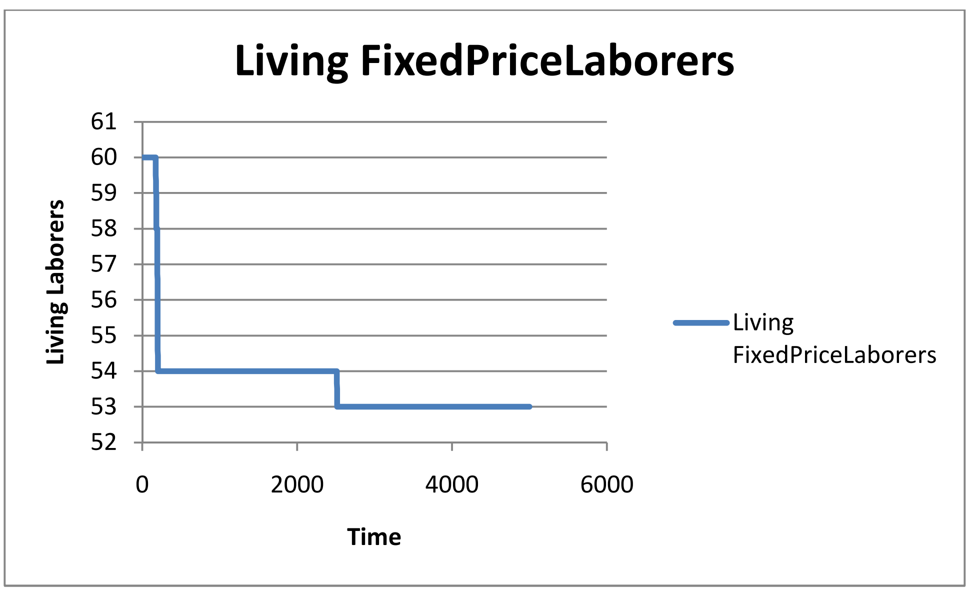

Figure 6B

Eventually, the farm became too poor to operate at maximum productivity. This caused some of the laborers to die of starvation.

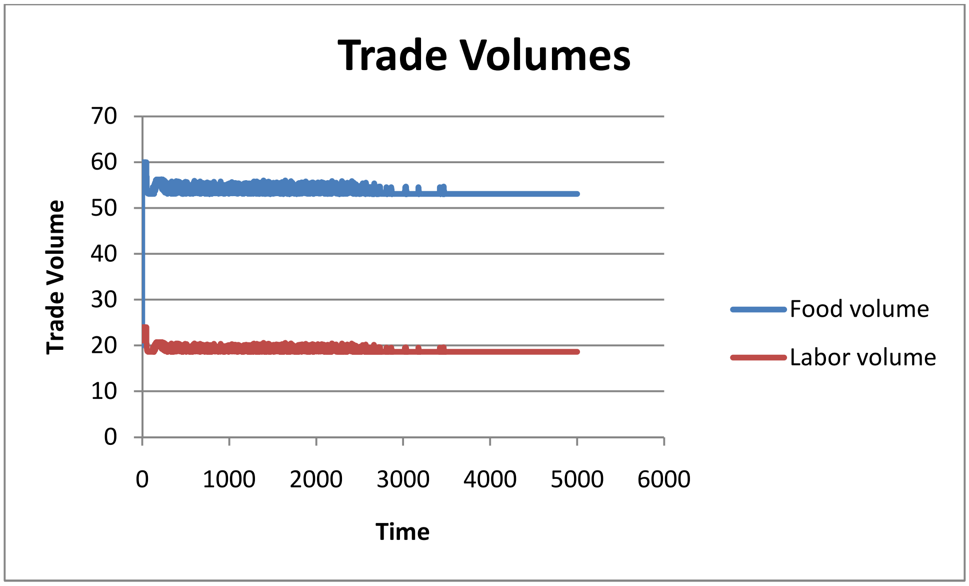

Figure 6C

Eventually, the laborer deaths reduced the food and labor volumes to new equilibrium values where the farm's monetary profit was zero.

7.

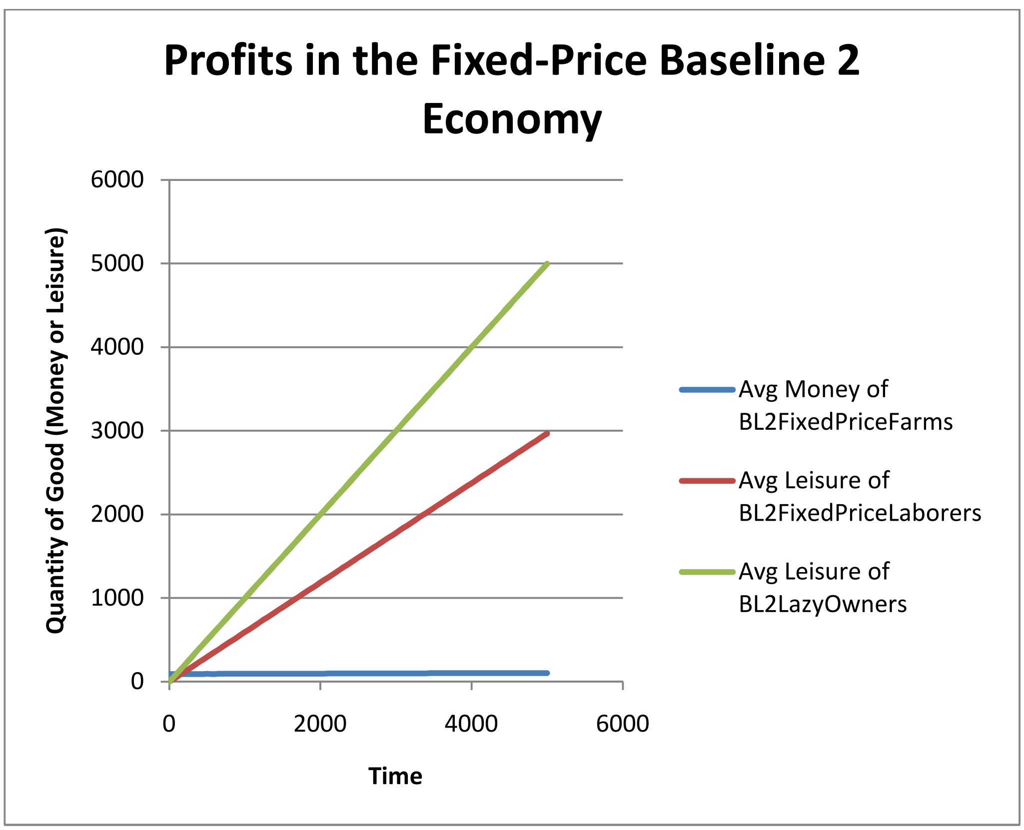

Figure 7

This plot shows the profits for each type of agent in the baseline 2 economy ($P_{leisure}$ for FPLs and LOs and P_{money} for FPFs). Both the FPLs and LOs steadily accumulate leisure, while the FPFs experience zero profit because their monetary profits are withdrawn by their owners, who then use them to earn leisure profits.

8.

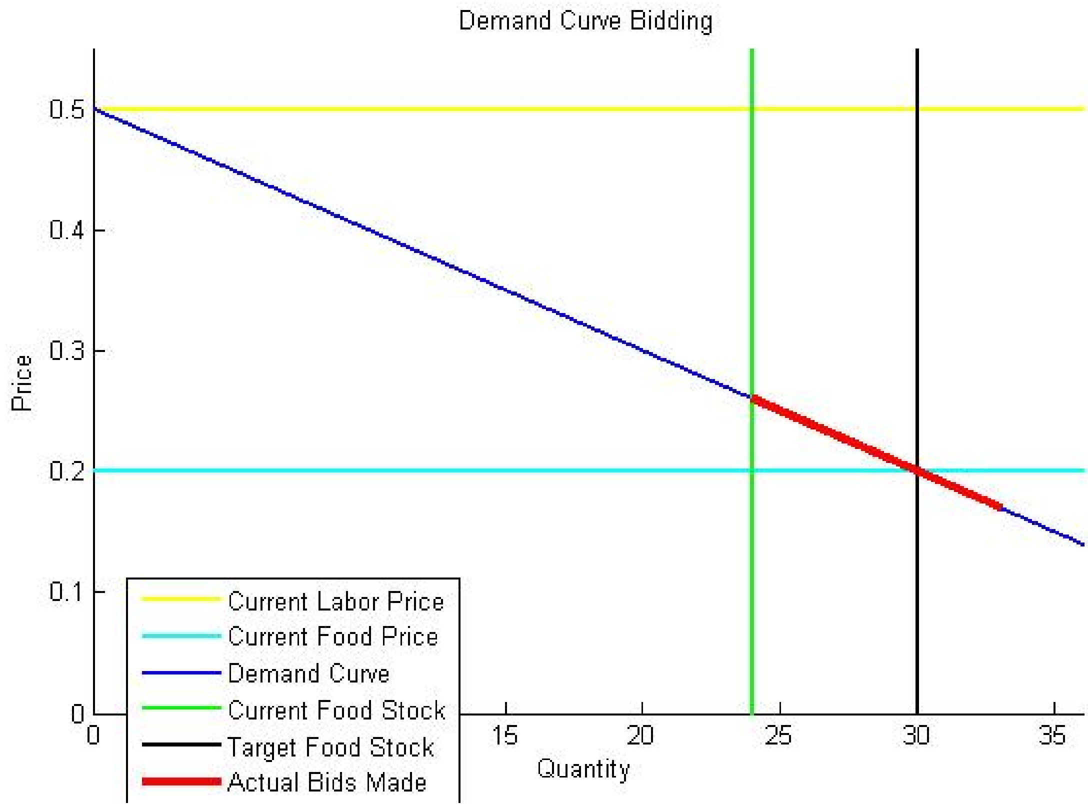

Figure 8

This plot shows the how current prices as well as a laborer's desired food stock are used by the demand curve bidding algorithm to determine a demand curve (blue). The laborer then makes several bids (red) along the curve, starting at his or her current food stock (green). This helps laborers maintain a comfortable stock of food, and provides mechanisms to drive the food price both up and down because the laborer makes bids both below and above the last market price.

9.

The following plots demonstrate the ability of demand and supply curve food-bidding strategies implemented in baseline 2 to adjust a dynamic food price to its "correct" value as determined by the stable price ratio.

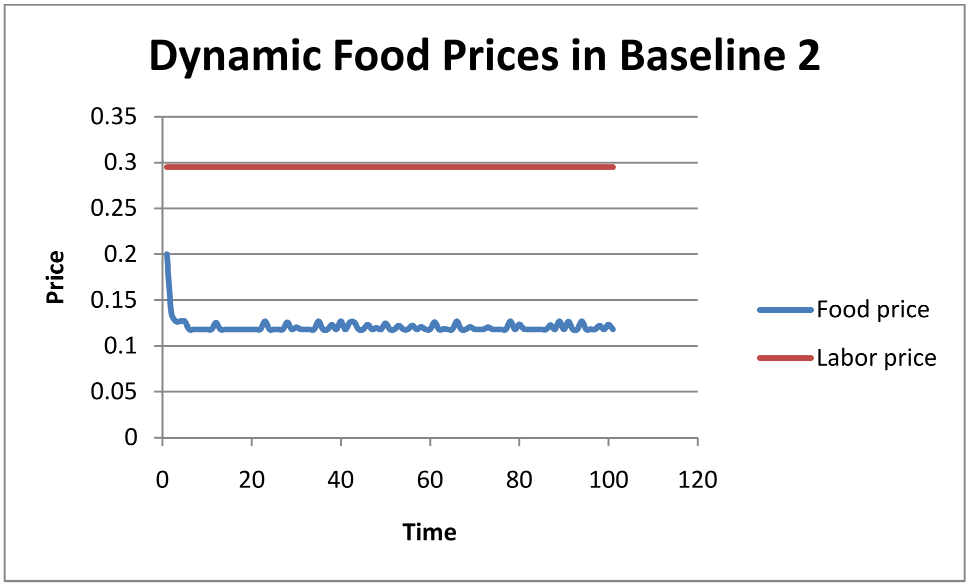

Figure 9A

In this simulation, the food price started at a value that was too high relative to the labor price. The stable price ratio predicts a stable food price of 0.122 for a labor price of 0.3.

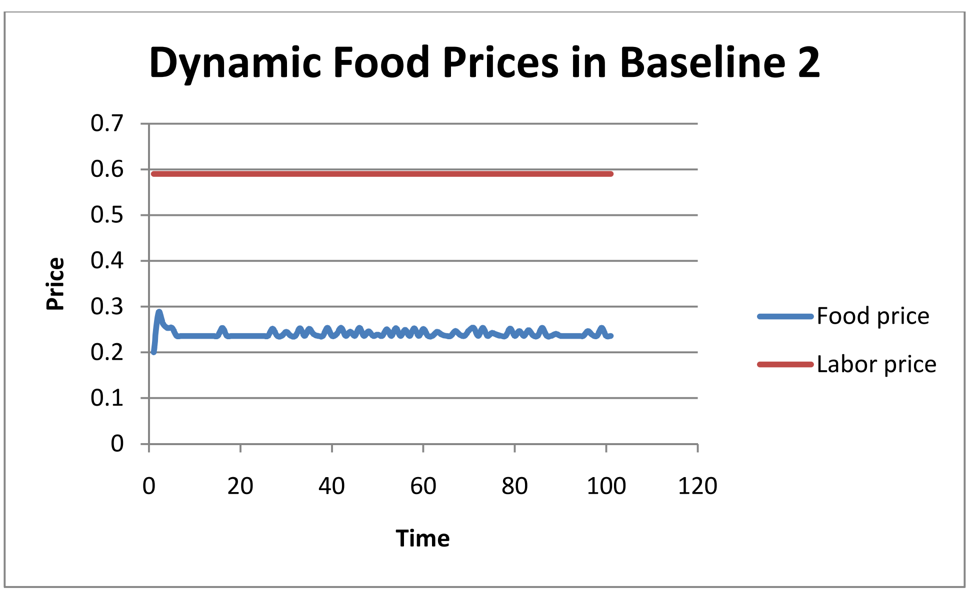

Figure 9B

In this simulation, the food price started at a value that was too low relative to the labor price. The stable price ratio predicts a stable food price of 0.244 for a labor price of 0.6.

10.

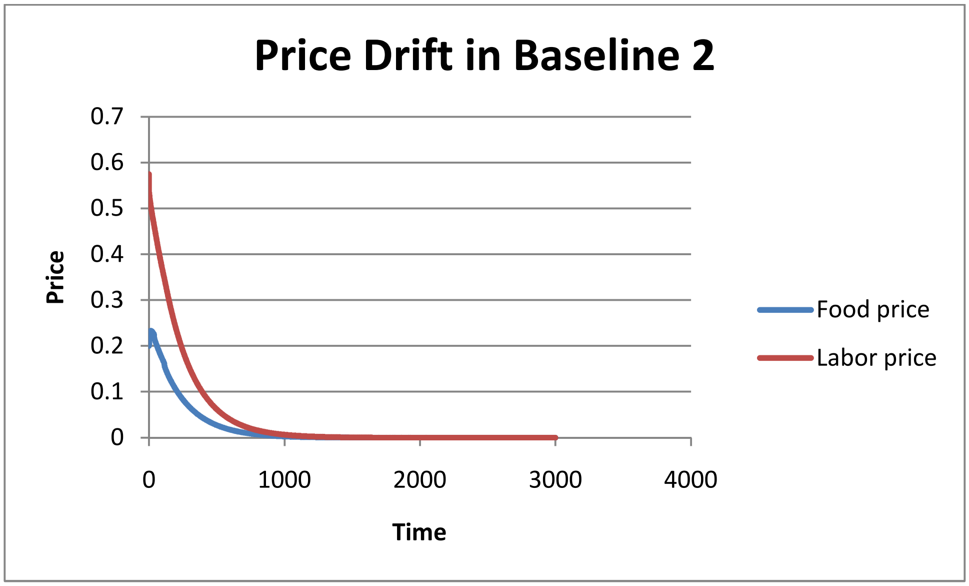

Figure 10

This plot shows an example of the price drift phenomenon that occurred when we allowed both prices to move freely. This simulation used a small population (< 75% carrying capacity), which caused the prices to continually fall.

11.

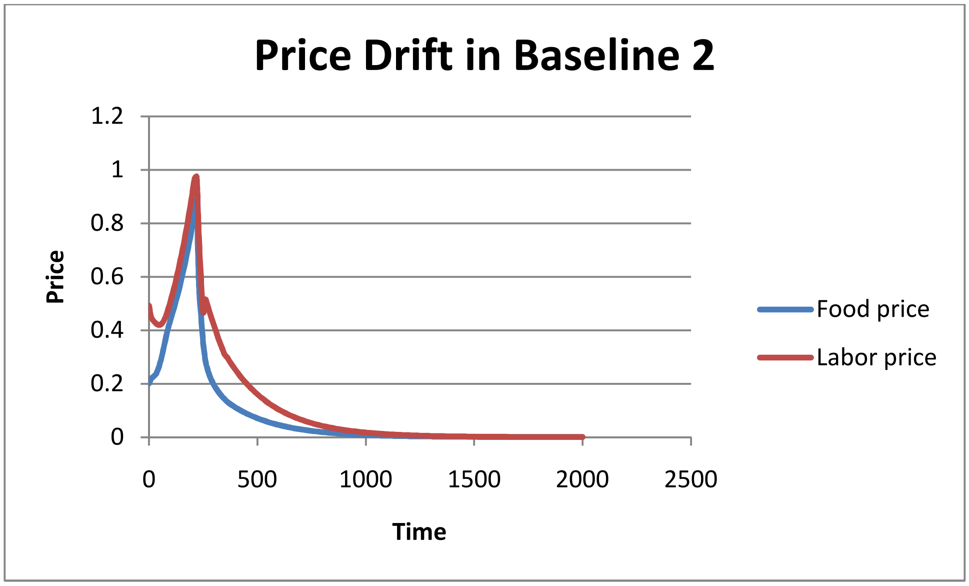

Figure 11

This plot shows an example of the price drift phenomenon that occurred when we allowed both prices to move freely. This simulation used a large population (> 75% carrying capacity), which caused prices to rise. Eventually, this trend led to the death of many laborers, which reversed the direction of the drift.

12.

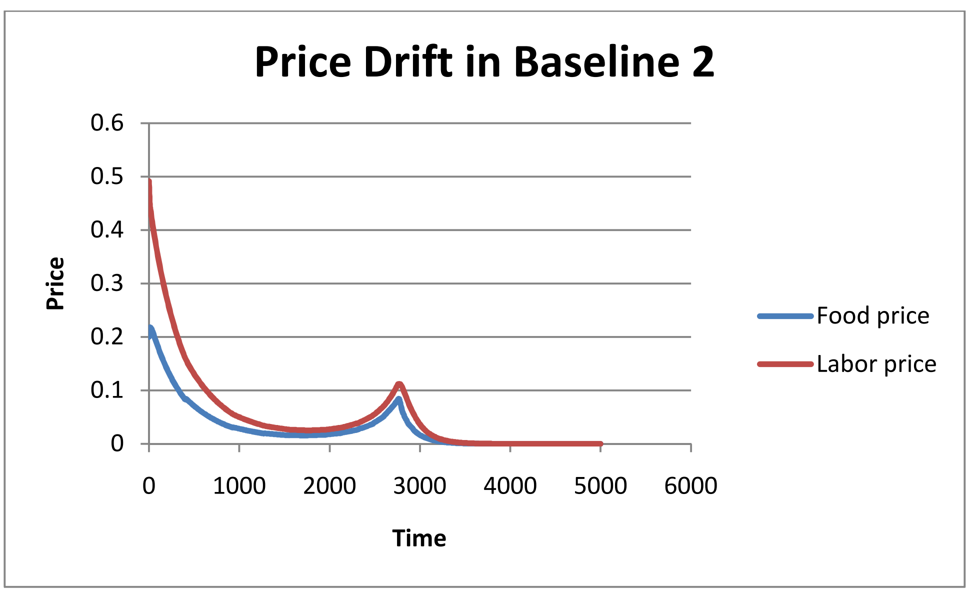

Figure 12

This plot shows an example of the price drift phenomenon that occurred when we allowed both prices to move freely. This simulation used a population size of about 75% carrying capacity. Around this size, both rising and falling trends were observable at different times. In the end, though, falling-price trend proved to be the stronger of the two.

13.

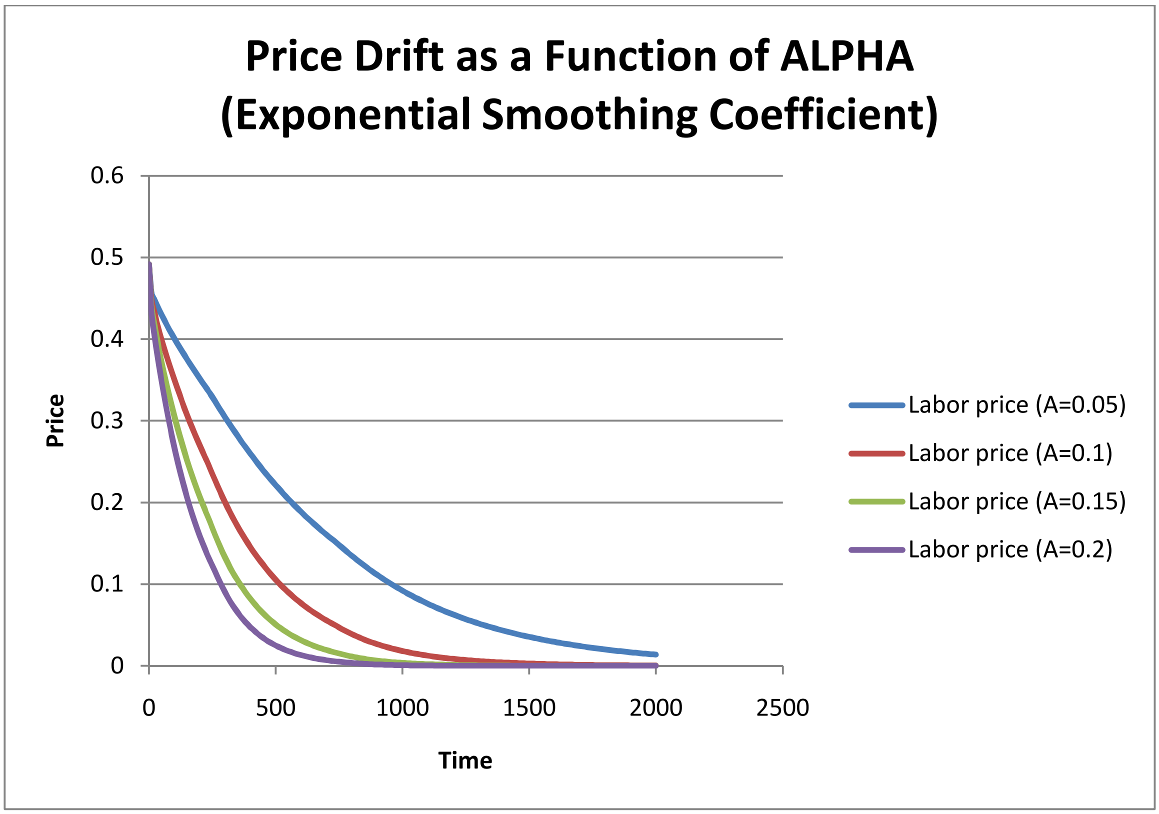

Figure 13

This plot shows the effect of exponential smoothing (incorporated into the agents' bidding algorithms) on price drift in the baseline 2 economy. As agents' give stronger weight to historical prices (lower α), the price drift slows, but the general trend does not change. Note that each curve represents a different simulation run.

14.

These plots (14A-B) show the characteristics of the quadratic production functions used in baseline 3.

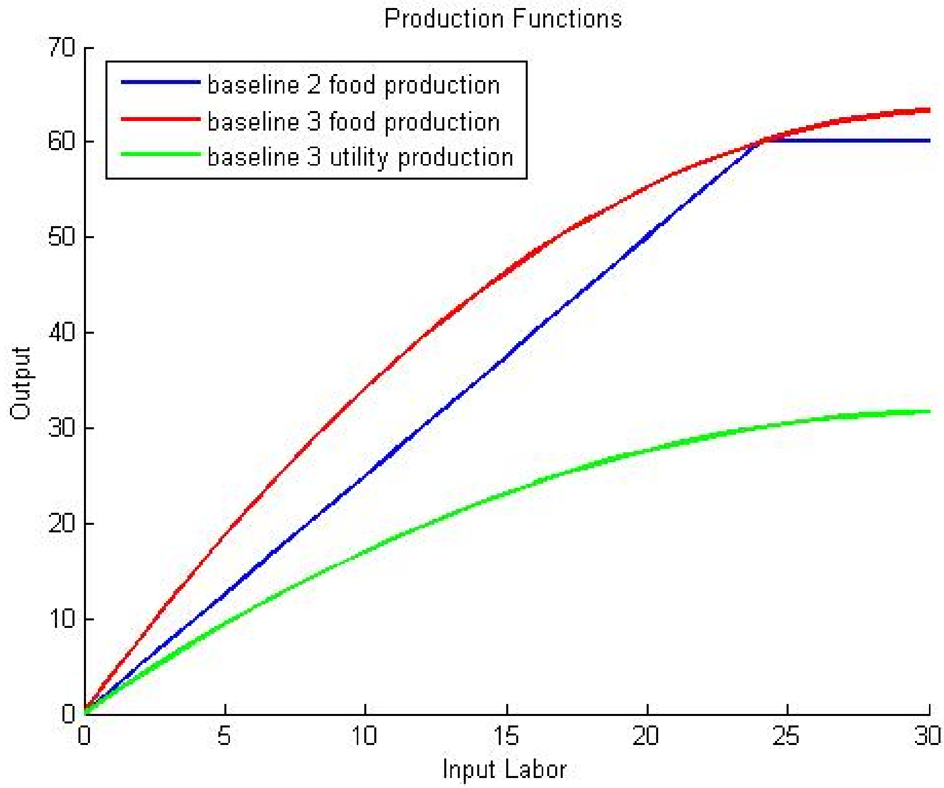

Figure 14A

The parameters of the baseline 3 food production were chosen to keep it similar to the capped linear function used in the baseline and baseline 2.

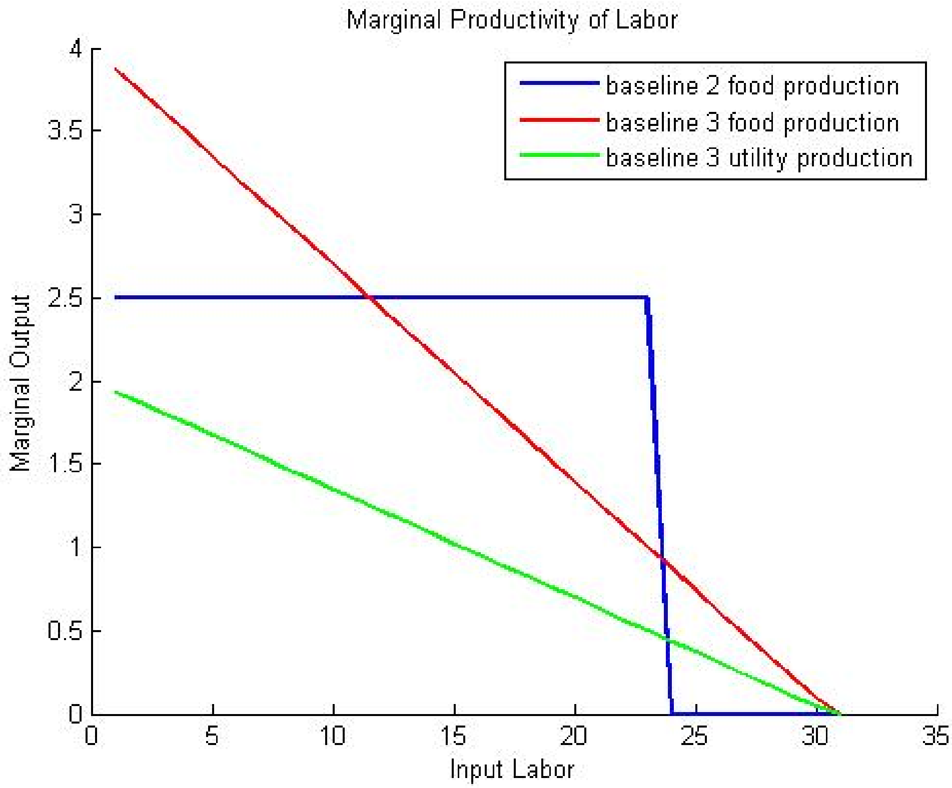

Figure 14B

This plot shows the marginal productivity of labor (MPL), which is simply the derivative of the production function. The linear, downward-sloping MPL curves of the quadratic production functions make it easy to quantify a firm's labor demand curve.

15.

These plots (15A-B) indicate the stability of the baseline 3 economy over a long simulation run.

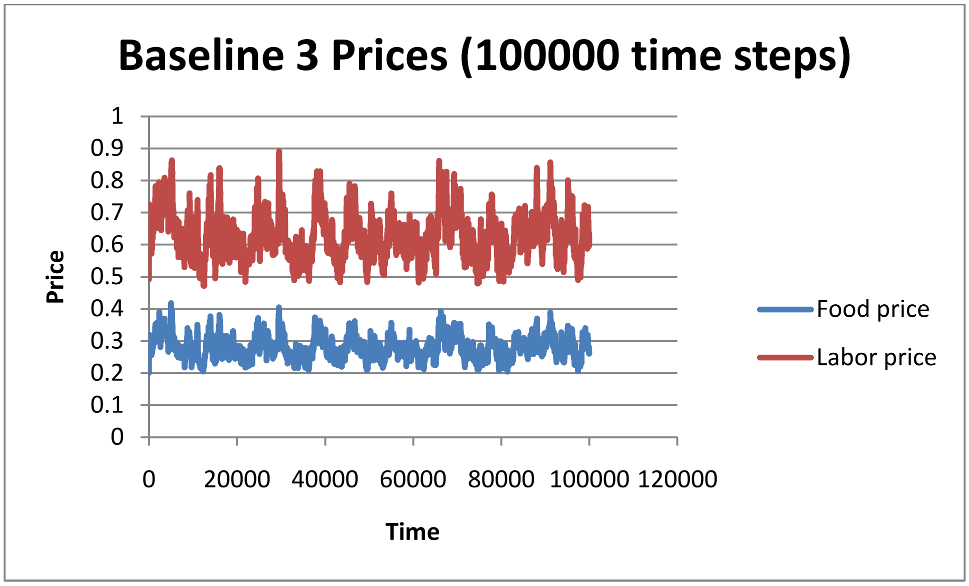

Figure 15A

Price change over time in this simulation is characterized by oscillations rather than price drift.

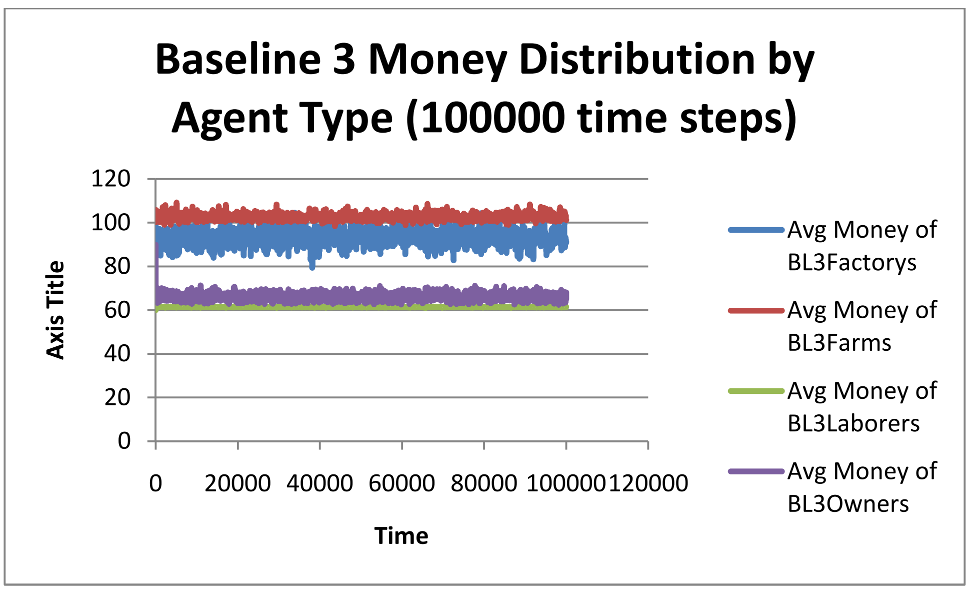

Figure 15B

No one type of agent in the baseline 3 economy manages to steadily accumulate money, even over 100000 time steps.

16.

These plots (16A-B) indicate the stability of the baseline 3 economy in a large simulation run.

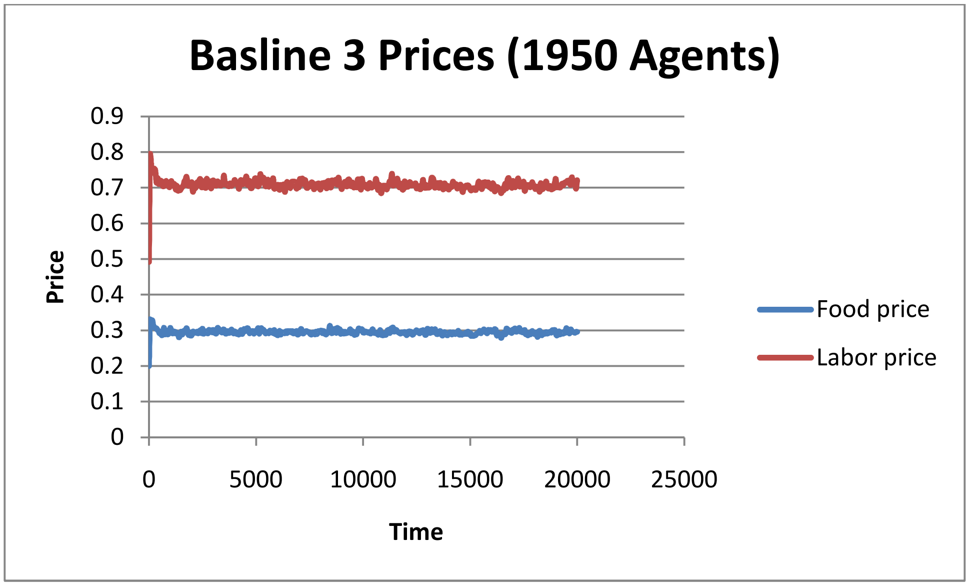

Figure 16A

Adding more agents to the baseline 3 economy appears lower the magnitude of price oscillations. This is not unexpected, since it seems reasonable that thickening the market (and thus lessening the effect of each agent's bids) would help to damp such movements.

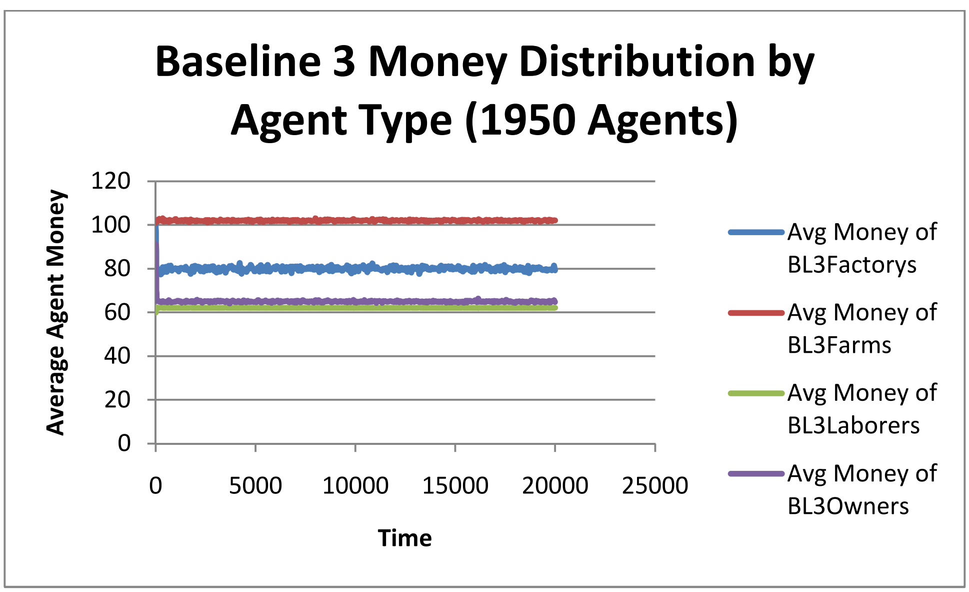

Figure 16B

In the large-population run one type of agent in the baseline 3 economy managed to steadily accumulate money.

17.

These plots (17A-D) demonstrate the results of an experiment where baseline 3 economies were simulated with very similar initial conditions to test whether the system could settle on different equilibria. Note that the values presented here are averaged across an entire simulation run.

Figure 17A

The average food and money prices vary greatly across multiple simulation runs. Note that each run used the same historical prices.

Figure 17B

However, the average price ratio (labor price : food price) varied only slightly. This could be an effect of starting with very similar initial conditions, or it may indicate something more fundamental about the baseline 3 model.

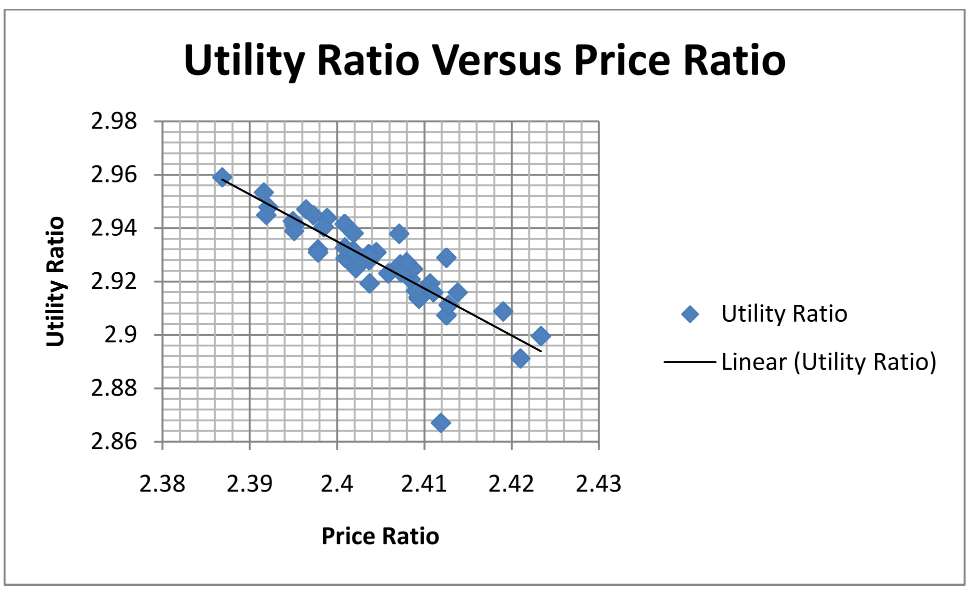

Figure 17C

The price ratio appears correlated with the utility ratio (average owner accumulated utility : average laborer accumulated utility). The utility ratio is an indicator of how the wealth created by the economy is distributed (in the form of utility) to its various members.

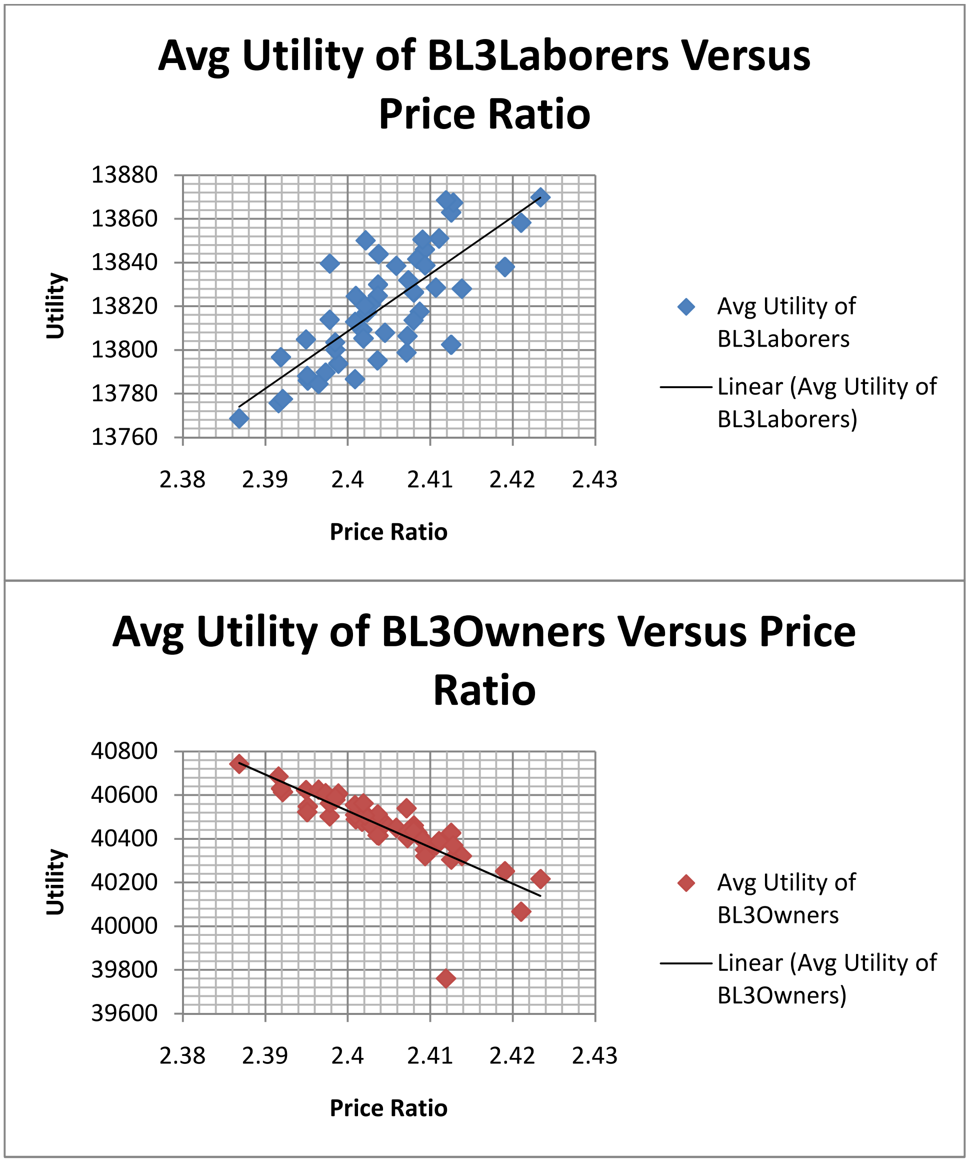

Figure 17D

Not only do laborers benefit relative to owners from a higher price ratio, they also benefit in an absolute sense. That is to say, higher price ratios do not just cause a reduction in the utility accumulated by owners; they actually result in laborers accumulating a greater amount of utility.

Further experimentation will be required to determine how price ratio affects total utility production across the entire economy, and whether the current baseline 3 model can stabilize at a point at which laborers and owners earn the same amount of utility.