Zero Knowledge Traders

Ernesto Carrella

March 30, 2016

Abstract

We provide simple general-purpose rules for agents to buy inputs, sell outputs and set production rates. The agents proceed by trial and error using PID controllers to adapt to past mistakes. These rules are computationally inexpensive, use little memory and have zero-knowledge of the outside world. We place these zero-knowledge agents in a monopolist and a competitive market where they achieve outcomes similar to what standard economic theory predicts.

1 Introduction

Agents are coordinated by the prices they set. Markets clear when prices balance correctly all possible economic information. Yet individual agents need to set these prices with only some of that information. I present a model where agents endogenously discover and set market-clearing prices using none of that information.

I give agents simple decision rules that allow them, with no knowledge of demand, supply or market structure, to solve for both competitive and monopolist prices and quantity. After a brief literature review in section 2, I explain how agents trade in section 3 and 4 of this paper. I then expand them to make the agents also produce and maximize profits in section 5 and 6 of this paper.

Agents using these rules are pure tinkerers. They adapt not by learning the real model of the world but by assuming such a model is unknowable and then proceeding by trial and error. Tinkering keeps these rules general-purpose.

I believe this methodology useful for two reasons. First, I provide a ready-made set of decision rules that can be used in almost any other agent-based model. There is no market structure or auctioneer feeding the prices to the agent. I believe them perfect as a baseline to compare to more nuanced decision rules.

Second, I provide a "rationality floor": the minimum information and rationality needed for markets to work, like the Zero-Intelligence project Gode and Sunder (1993). Unlike Zero-Intelligence, my rules apply to both trading and production and do not depend on the very strict statistical assumptions that doomed Zero-Intelligence Cliff et al. (1997).

The lack of knowledge assumed in this model is extreme to the point of caricature. This is by design. Firstly because the fewer informational assumptions I make, the easier it is to plug-in these rules in other models. Secondly because I can test the robustness of traditional partial equilibrium analysis to a complete violation of standard rationality assumptions.

As Mäki (2008), but see also Nowak et al. (2011), I claim that the fundamental contribution of any model is to isolate causal mechanisms in a complicated world. Here the mechanism allows firms to maximize profits and price goods correctly just by monitoring the difference between what they produce and what they sell.

In a more realistic model, firms would have more information and intelligence but they would need to solve a higher dimensional problem trying to manage not just production and prices but also customer satisfaction, labor relations, geography, social networks and so on, mixing all causal mechanisms in a single incomprehensible cacophony of parameters. That model would resemble reality better but it wouldn't be more useful.

2 Literature Review

I can categorize market processes along two axes. First whether the price vector is provided exogenously or discovered endogenously. Second whether the process allows trades to occur in disequilibrium before the equilibrium price is found.

Exogenous-equilibrium: the modeler solves for the market clearing prices and assumes they are known to the agents. This requires agents to be as rational, informed and computationally capable as the modeler who created them. Unfortunately the computational ability assumed is very high: even when equilibrium prices are known to exist, the utility is linear and the goods are indivisible, approximating equilibrium prices is NP-hard Deng et al. (2002). More generally exchange equilibrium is a PPAD(Polynomial Parity Arguments on Directed graphs)-complete problem Papadimitriou (1994).

Exogenous-disequilibrium: the modeler imposes a market formula that changes prices in reaction to some aggregate variables like excess demand or productivity. Scarf's paper on the subjectScarf (1960) is instructive in both explaining the idea and giving examples where an equilibrium exist but this methodology fails to find it. When finding an equilibrium is not an explicit goal, agent-based models use this methodology: for example the wage-setting algorithm in Dosi et al. (2010).

Endogenous-equilibrium: the agents play a game against one another (for example a Bertrand competition) and choose the Nash equilibrium price and quantity. This still requires agents both to have enough information about their competitors to feed into their best response function and high computational ability: finding Nash-equilibria is also PPAD-complete Chen and Deng (2006).

Endogenous-disequilibrium: agents interact and trade between themselves without waiting or solving for equilibrium. The oldest market process model, the Walras's tatonnement, involved independent agents exchanging tickets at

disequilibrium prices (see chapter 3 of Currie and Steedman (1990)). Agents traded tickets rather than goods because trading goods at disequilibrium creates wealth effects and path dependencies that invalidate welfare theorems; see Jaffe (1967) for a discussion about Walras, see Foley (2010) for a modern treatment on welfare theorems under disequilibrium. Modern disequilibrium models usually don't assume welfare theorems hold. I catalog these models by the market structure used.

In a strictly bilateral market, agents are randomly matched and barter with one another. There is no single market price but many trade prices. The pricing strategy depends on the matching and bartering functions used. If agents can compare their profitability with the rest of the population, like in Gintis (2007), the prices offered by each agent can be driven by evolutionary methods. If an agent only knows the characteristic of whom it is matched to, like in Axtell (2005), market clears by letting every beneficial barter occur between all trader pairs. Results can be driven by matching rather than bartering, as in Howitt and Clower (2000), where fixed-price shops are built endogenously and agents have to search for the right shops to exchange goods.

A more general market structure is the continuous double auctions with multiple buyers and sellers. My model belongs to this category. Here the two main behavior algorithms are: Zero Intelligence Plus Cliff (1997) and the Gjerstad and Dickhaut method Gjerstad and Dickhaut (1998). They represent the two opposite views on adaptation: tinkering and learning. Zero Intelligence Plus traders tinker with their markup according to the previous auction results while Gjerstad and Dickhaut auctioneers first learn a probabilistic profit function and then maximize it.

My algorithm is simpler. Like Zero Intelligence Plus, I set prices by tinkering over previous errors. Unlike Zero Intelligence Plus, I use no auction-specific information and so my algorithm is market-structure independent. Moreover my algorithm can be expanded to direct production and maximize profits rather than just trade. The tinkering and adjustment is simulated through the use of Proportional Integral Derivative (PID) controllers.

While control theory is a staple of macroeconomics and PID controllers the simplest and commonest of controls, to the best of my knowledge I am the first to use PID control in economics. The closest paper to my approach is Ortega and Lin (2004) where a PID controller is suggested for inventory control, equalizing new buy orders to warehouse depletion. That is not an economic model as it doesn't deal with prices or markets. In the spirit of Bagnall and Toft (2006), I judge my algorithm by testing it in a series of markets where the economic theory identifies a clear optimum.

3 Zero-Knowledge Sellers

The seller is tasked to sell 100 units of a good every day. It has no information on demand or competition and no opportunity to learn. All the agent can do is set a sale price and wait. If at the end of the day it has sold too much, it

will raise the price tomorrow. If it has sold too little, it will lower the price. This is an elementary control problem. The seller has a daily target of 100 sales and wants to attract exactly 100 customers a day. The seller has no power over customers themselves and so it needs to manipulate another variable (sale price) to affect the number of customers attracted. The seller doesn't know what the relationship between sale price and customers attracted is and so proceeds by trial and error. The trial and error algorithm used by sellers in this paper is a simple PID controller.

Given target y^* (target sales) and process variable y (today's number of customers), the daily error is:

$$e_t = y_t^* - y_t \tag{1}$$

Define u_t as the policy (sale price). The seller manipulates the policy in order to reduce the error. The true relationship between policy and error is unknown, so the seller follows the general rule: "increase the policy when the error is positive, decrease it when the error is negative", which is the definition of negative-feedback control Åström and Hägglund (2006).

The PID controller manipulates the policy as follows:

$$u_{t+1} = ae_t + b\int_0^t e_\tau d\tau + c\frac{de_t}{dt} \tag{2}$$

Intuitively the policy is a function of the current error (proportional), all observed errors (integral) and the change from the earlier errors (derivative). In discrete time models (as the simulations in this paper) the equivalent formula is:

$$u_{t+1} = ae_t + b\sum_{i=0}^{t} e_i + c(e_t - e_{t-1} \tag{3}$$

There are four reasons as to why PID controllers are a good choice to simulate agents' trial and error. First, PID controllers assume no available information. Agents using PID rules act only on the outcome of previous choices.

Second, PID controllers assume no knowledge on how the world works. The PID formula contains no hint on how policy affects the error: there are no demand or supply functions. The PID formula leads agents to tinker and adapt without ever knowing or learning the "true" model.

Third, PID controllers make no assumptions on what the target should be. The target in the PID formula is completely exogenous. The controllers work regardless of how the target is chosen or how often it is changed.

Finally, PID controllers can complement other rules through feed-forwarding. Feed-forwarding refers to using PID controllers on the residuals of other rules. For example, take a more nuanced seller choosing its sale price by estimating a market demand function from data. This estimation would provide an approximate prediction of demand given the sale price. Still we could improve this approximation by adding a PID controller to adjust the sale price by setting as error the discrepancy between predicted and actual demand.

For PID controllers to work, four assumptions need to be made on the market in which they are employed. First, PIDs work by trial and error so the market structure must allow agents to experiment. This means that policies (prices) must be flexible. The stickier the policies, the slower the agent is at zeroing the error.

Second, PID controllers work better when even small changes in policy have some effects on the error. For Zero-Knowledge sellers the equivalent assumption is facing a continuous demand function. This does not mean that discontinuities automatically invalidate PID control and in fact all the computational examples in this paper have discrete and discontinuous demands. But PID performance degrades with discontinuities resulting in more overshooting and slower approach to the equilibrium prices.

Thirdly, PID controllers implicitly assume a downward sloping demand: lower prices increase sales, higher prices decrease them. Zero-Knowledge sellers would fail to price Giffen goods.

Finally, targets must be achievable. For example, finding the price to sell to exactly n agents in a world with infinitely elastic demand is impossible. The target "exactly n sales" is unreachable: the error will oscillate between n and infinity, never reaching zero. Section 5.2 of this paper deals with how to set targets endogenously.

4 Zero-Knowledge Sellers Example

4.1 Mathematical Example

It is possible to show the workings of a Zero-Knowledge seller without software. Take a seller facing the unknown demand curve and tasked to sell 5 units of good every day. This Zero-Knowledge seller uses a PID controller with the parameters 0.01 for the proportional error, 0.15 for the integral and 0 for the derivative. Table 1 tracks the trial and error process of the seller as it discovers the right price (19) and sells the right number of goods.

Table 1: Non-Computational Example of a Zero-Knowledge Seller

Day	e_t	$\sum_{i=0}^{t} e_t$	Price (u_t)	Quantity to sell (y_t)	Customers Attracted
1	·	·	0	5	100
2	95	95	15.2	5	24
3	19	114	17.290	5	13.550
4	8.55	112.55	18.468	5	7.660
5	2.660	125.210	18.808	5	5.960
6	0.960	126.170	18.935	5	5.325
7	0.325	126.494	18.977	5	5.113
8	0.113	126.607	18.992	5	5.039
9	0.039	126.646	18.997	5	5.005

4.2 Computational Example

A seller receives daily 4 units of a good to sell. There is a fixed daily demand made up of 10 buyers. The first is willing to pay $90 or less for one good, the second $80 and so on. The demand repeats itself every day. Agents trade over an order book: the seller sets its price and all crossing quotes are cleared (while supplies last). The trading price is always the one set by the seller. Prices can only be natural numbers. The demand-supply schedule is shown in figure 1.

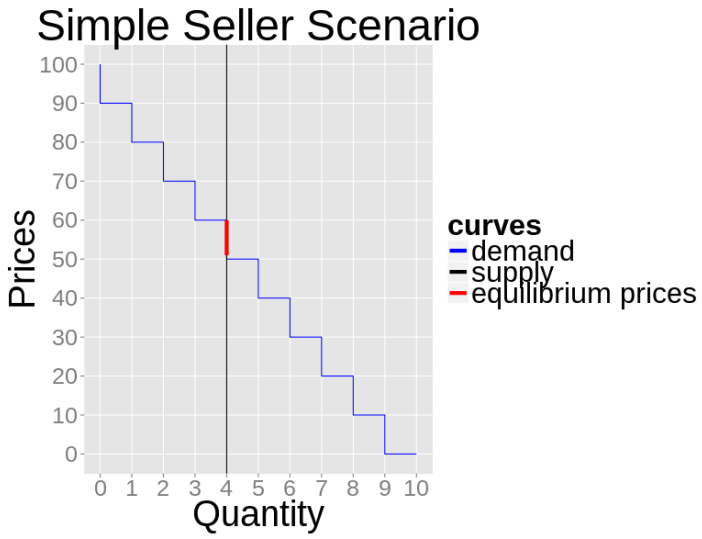

Figure 1: The example's daily market demand and supply

The seller starts by charging a random price and then adjusts it daily through its PID. The seller target is to sell all its inventory. Unsold goods accumulate. The seller knows only how many customers it attracted at the end of the day. There is no competition.

With this setup, any price between $51 and $60 (both included) will sell the 4 goods to the 4 top-paying customers. Figure 2 and figure 3 show the market closing prices of two sample runs. In both cases the seller selects the "right" price: $51.

Notice how, when the initial price is too high as in figure 3, the adjustment initially undershoots. Undershooting is caused by the firm trying to dispose leftover inventory from previous days; that is, while undershooting, the firm is trying to sell its usual 4 daily goods plus what has not been sold before.

4.3 PID Parameter Sweep

The PID equation depends on three parameters. Parameter a for the proportional error, b for the integrative error and c for the derivative one. In the

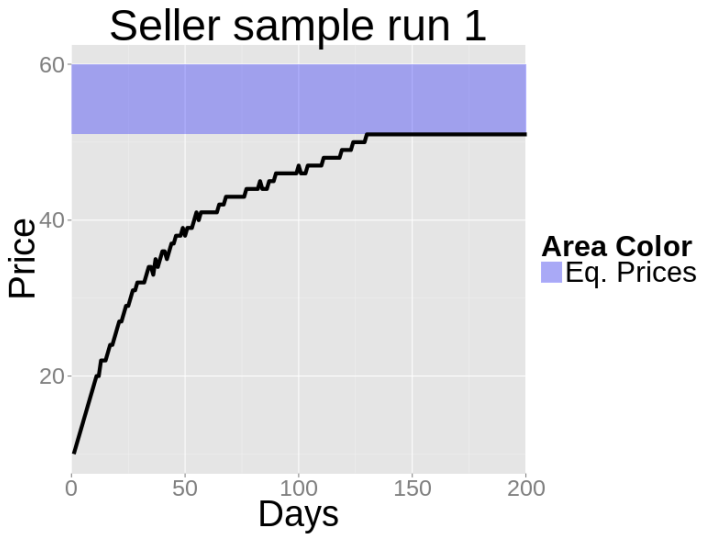

Figure 2: The closing prices of a Zero-Knowledge seller sample run when the initial random price is below the equilibrium

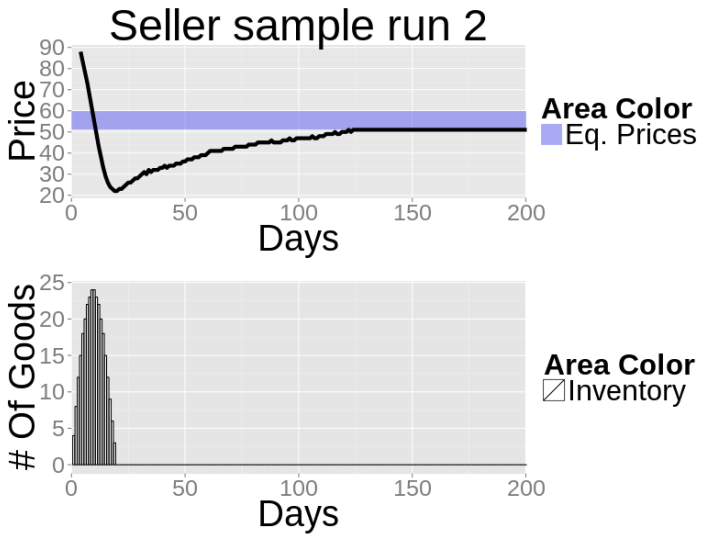

Figure 3: The closing prices and inventory of a Zero-Knowledge seller sample run when the initial random price is above the equilibrium

previous example the parameters were $a = 0.25$, $b = 0.25$ and $c = .0001$. Here I vary the parameters in turn to show their effects on sellers' behavior.

In figure 4 the a parameter varies. An increase in a makes the PID more responsive to today's error. This does not results in a faster approach to true prices but only a more jagged price curve.

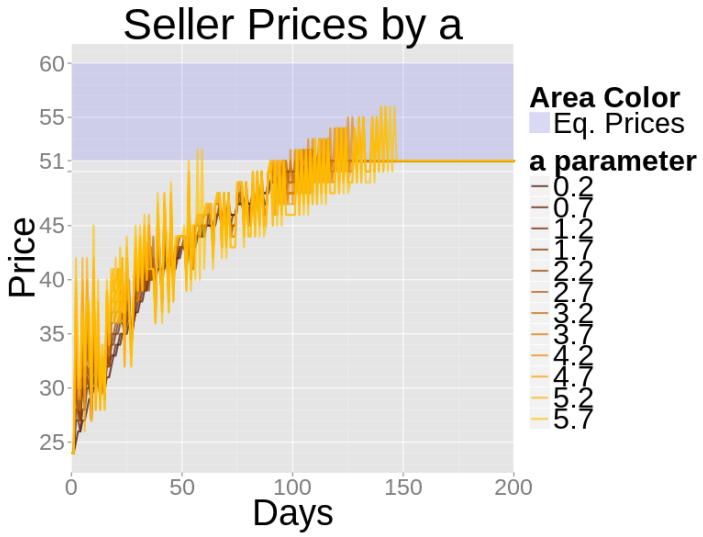

Figure 4: The effects of varying the a parameter of a Zero-Knowledge seller

In figure 5 the b parameter varies. An increase in b makes the PID more responsive to the cumulative sum of errors. This results in a faster approach to the true prices but it can cause fluctuation and overshooting.

Changing the c parameter (even increasing it by 100 times) has almost no effect in this model. The derivative part of the PID becomes important to smooth overshooting which isn't a real issue to Zero-Knowledge sellers because their baseline parameters are very small.

4.4 Computational Example with Demand Shifts

Agents using PID controllers adapt rather than learn. This keeps them working when market conditions change. Here I replicate the Zero-Knowledge computational example of the previous sections but after 500 days 10 more buyers enter the market. These buyers have a higher demand: the first willing to pay \$190, the second \$180 and so on. The "right" price, after the shock, moves from \$51 to \$151.

Figure 6 shows the sale prices of a Zero-Knowledge seller. The seller quickly finds the new price. Notice here that nothing was changed in the seller algorithm. The PID was not told that the demand had shifted. There is no

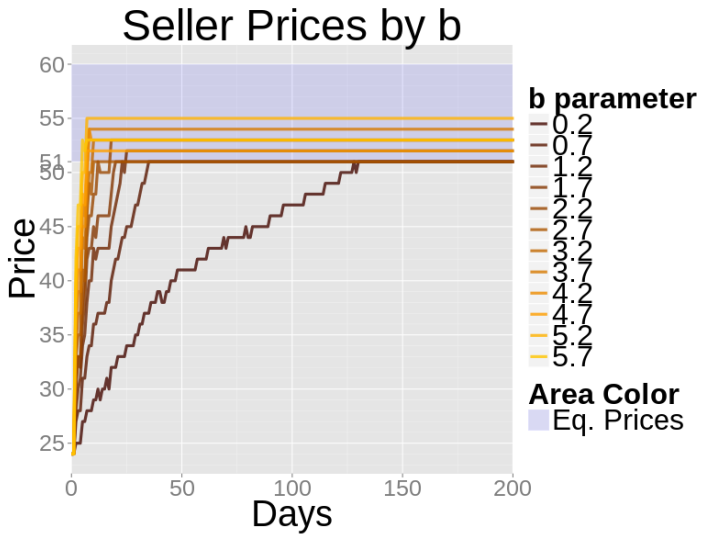

Figure 5: The effects of varying the b parameter of a Zero-Knowledge seller

"structural break" detection. Simply the PID reacts to a changing y (number of customers) by increasing prices to hit the old target.

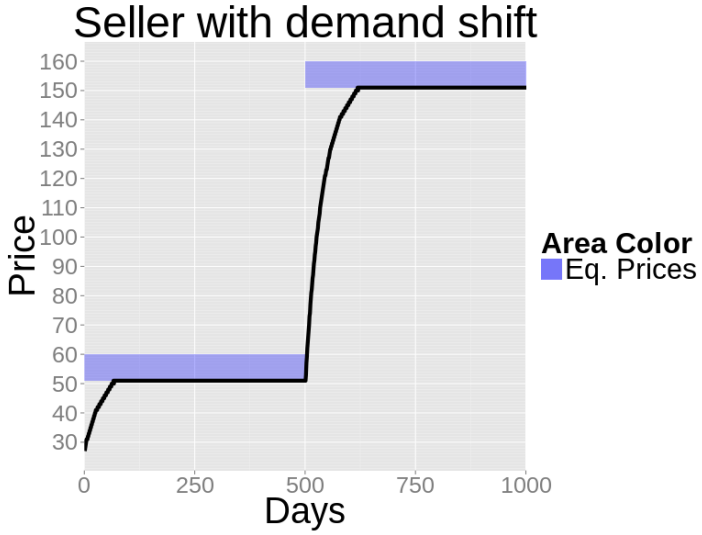

Figure 6: The sale prices of a Zero-Knowledge seller dealing with a demand shock after 500 days

5 Zero-Knowledge Firms

A firm is tasked to maximize its profits by producing and selling its output daily. It has no information on customer demand, labor supply or competition. The firm only knows its own production function. The firm has to decide daily and concurrently the sale price of its goods, the wage of its workers and its production quotas. The problem faced by the firm is harder for two reasons: firstly, it has to trade in multiple markets at the same time and secondly, it is a producer, not a passive receiver of endowment.

Zero-Knowledge firms maximize their profits by dividing the problem into sub-components and solving each separately. There are two equivalent ways to understand this division: by variables or by time as in figure 7.

Dividing the profit maximization problem by variables means recognizing that the firm has two kinds of variables to set:

- **Targets**: how much to produce, how much input to buy, how much output to sell, how many workers to hire. See figure 9
- **Policies**: how much to offer for inputs, how much to ask for outputs, what wages to offer. See figure 8

Rather than setting them all together at once, we proceed in turn. We manipulate policies in order to achieve targets, and we set targets in order to maximize the profit function.

The process of profit maximization of the Firm then is split in two classes of operations:

- Control: change policies to achieve targets
- Maximization: change targets to achieve the objective.

The alternative and equivalent way to subdivide the profit maximization process is by focusing on time. Here I take Hicks's Leijonhufvud (1984) division of time in economics between long run (both capital and labor are variable), short run (labor is variable) and market days (production is fixed and unchangeable). Control is the process of managing Hicksian market days: buying, hiring and selling assuming production can't be changed. Maximization is the process of managing the short run: changing production rate to maximize profits.

The two processes integrate as a feedback loop. The maximization process sets production targets for the controls. The controls, given time, discover the price associated with those targets. The maximization then uses the discovered prices to adjust to new targets and the loop restarts. This is a trial and error alternative to proper backward induction. Backward induction requires the firm to try every possible target, discover the prices associated with each and then choose the target that maximizes profits. Backward induction is exhaustive learning, while the maximization used by the Zero-Knowledge firm is just tinkering.

An example of how the two processes relate in time is shown in figure 10. Control happens every day while maximization occurs less frequently to give

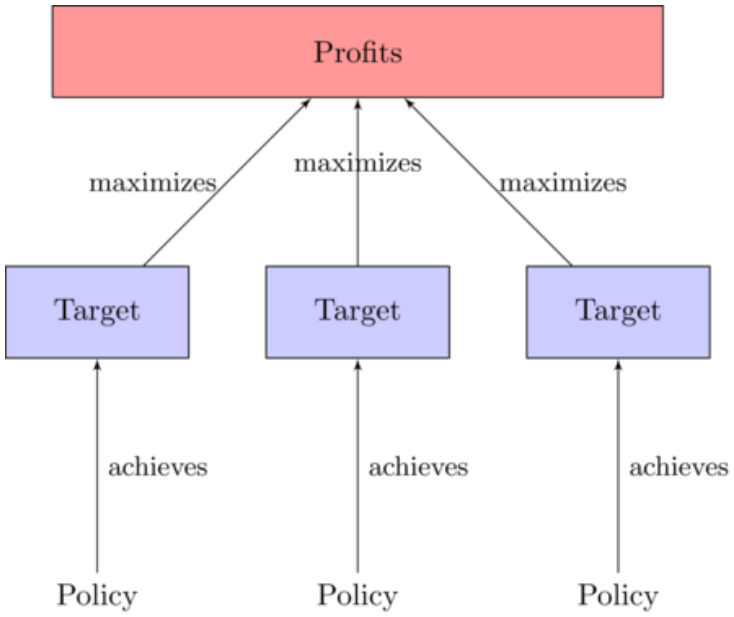

Figure 7: The sale prices of a Zero-Knowledge seller dealing with a demand shock after 500 days

control time to discover the right prices. In this example the firm revises its production every 3 days. This frequency is arbitrary, and in fact when and how one temporal phase ends and another begins has always been a weakness of Hicks's temporal model Currie and Steedman (1990). I will show in the Section 5.2 how to avoid this arbitrariness and link the frequency of maximization with the results from the control process.

5.1 Control

Control is the process of manipulating policies to achieve targets. In section 3 I used a PID controller to solve a univariate problem: manipulate one policy to achieve one target. The Zero-Knowledge firm problem is multivariate as it needs to manipulate the prices for all outputs and all inputs.

I solve this multivariate control problem by splitting it into multiple independent univariate control problems. The Zero-Knowledge firm is composed of many Zero-Knowledge traders each achieving a single target with their own PID controller. I call each of these traders a firm's department. This structure is appropriate for object-oriented programming through simple object composition, see figure 11.

In section 3 I showed how the PID controller solves the seller problem. Table 2 expands the PID methodology to buying and hiring.

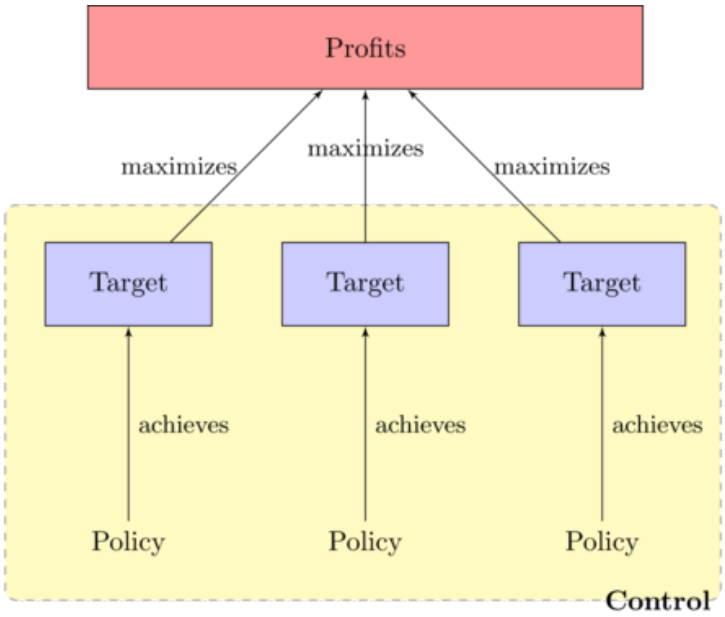

Figure 8: Define control as the process of changing policies to achieve targets

Table 2: PID variables for each firm department

Component	Variable y	Target y^*	Policy u_t
Purchases	# of goods purchased	# of input needed	Price offered
Sales	# of customers	# of output produced	Price demanded
Human Resources	# of workers	Target # of workers	Wage offered

If the firm produces more than one kind of goods, then it will have more than one sales department, each focusing on one kind of output.

I chose here to have departments targeting and dealing with flows rather than stocks thus reducing the need for inventory management and so making PID controllers simpler to use and less sensitive to the parameters I set. Focusing on stocks is not impossible, but it is harder and requires more tuning of the a and b parameters to keep the same level of accuracy Smith and Corripio (2005).

5.2 Maximization

Maximization is the process of finding the targets that maximize profits. From section 5.1 I know that the firm uses its controls to discover the prices (and therefore the profits) associated with a specific target. The maximization process involves adjusting targets given the information discovered by the control.

For control problems, I used the PID algorithm to adjust policies. I can't use

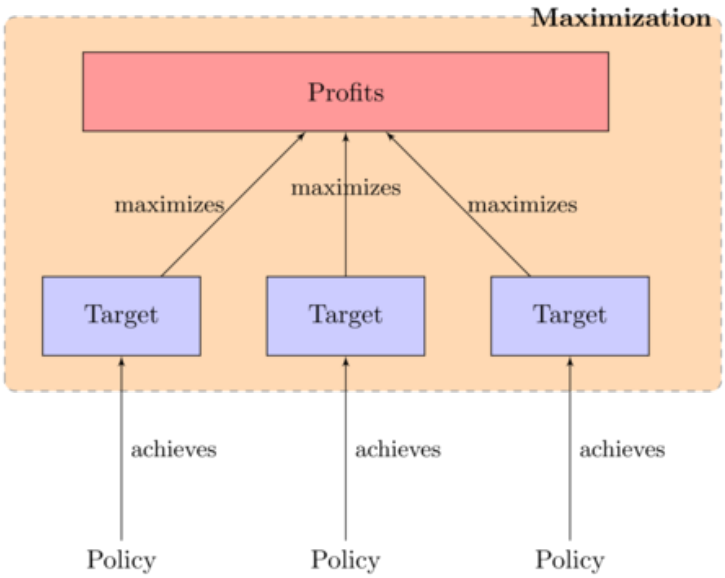

Figure 9: Define maximization as the process of setting target to maximize profits

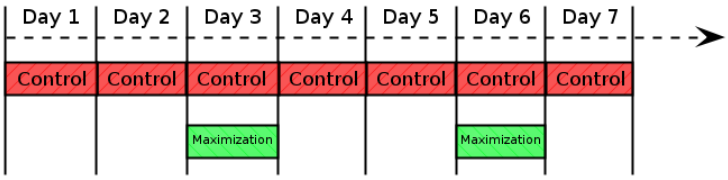

Figure 10: An example of how control and maximization processes occur over time. In this case a firm arbitrarily revise its production quota every 3 days, hence maximization on day 3 and 6. The firm needs to buy inputs and sell output every day, hence control every day of the week

a PID to adjust targets since the error (which in this case would be the distance from maximum profits) is unknown (the firm doesn't know what the maximum profits are). Therefore I use a more rudimentary adjustment algorithm.

I proceed in three steps. First, I simplify the maximization problem from multivariate (many targets to set together) to univariate. Second, I show the adjustment algorithm used to choose this target over time. Third, I define how much time the maximization algorithm should give to controls to discover the prices associated with each target.

Mathematically I want to maximize the profit function $\Pi(\cdot)$ by setting the

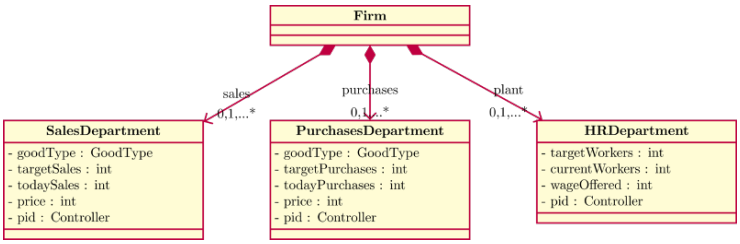

Figure 11: UML Diagram of a generic firm

vector targets y^* :

$$\max_{y_1^*, y_2^*, \ldots, y_n^*} \Pi(y_1^*, y_2^*, \ldots, y_n^*) \tag{4}$$

This is a multivariate maximization where each variable is the target of an independent control process. I want to avoid having to explore the whole combination space to find the right target vector, so I am going to condition the targets among themselves to reduce this maximization to a single variable. The main target the firm sets is the number of workers to hire L; this is equivalent to setting the daily production quota $f(L)$. I then set sales targets equal to production (sell everything you make) and buying targets equal to daily inputs (buy everything you need). The maximization problem becomes:

$$\max_{L} \Pi(L) \tag{5}$$

I use algorithms 1 and 2 to adjust targets after observing profits. Both algorithms are simple hill-climbers to show that no special maximization is required. Both algorithms use little memory, choosing the new workers' target based only on the present and the previous one.

Algorithm 1 Simple One-Shot Hill Climber maximizer

```
L ← 0                                          ▷ Start by having no workers
loop
    oldProfits ← Π(L)                          ▷ remember the current profits
    L ← L + 1                                  ▷ Increase worker size
    wait                                       ▷ Wait for controls to adapt
    if Π(L) < oldProfits then
        L ← L − 1                              ▷ Step back one, this is our final maximum
        break
    end if
end loop
```

Algorithm 2 Forever Hill Climbing maximizer

```
1:  L ← 0                                   ▷ Start by having no workers
2:  d ← 1                                   ▷ We start with positive direction
3:  loop
4:      oldProfits ← Π(L)                   ▷ Remember the current profits
5:      L ← L + d                           ▷ Tweak the worker force
6:      wait                                ▷ Wait for controls to adapt
7:      if Π(L) < oldProfits then
8:          d ← −d                          ▷ Continue in the opposite direction
9:      end if
10: end loop
```

Line 5 in algorithm 1 and line 6 in algorithm 2 expects the command "wait". This is because controls need time to change policies to achieve the new targets. The wait time can be arbitrary (e.g. one week, one month), but I found it more natural to make it conditional on control achieving targets (e.g. a week after all targets have been achieved). Conditional wait time has the advantage of heterogeneity so that different firms with different controls can use the same maximization algorithm at different frequencies. It is also how I endogenously connect the Hicksian "market days" and "short run" that is the relative speed with which agents change prices and change production targets.

Like with control, having Zero-Knowledge has drawbacks. There are two major drawbacks with this maximization procedure: an economic problem and a practical one.

Trial and error maximization is economically inefficient. Until the profit maximizing targets are found, the firm spends time either under or over-producing. This performance can influence the decision and profitability of suppliers, clients and competitors which are also groping for the right targets. In a Zero-Knowledge setting one agents mistakes can have externalities through the rest of the system.

This maximization is also susceptible to noise due to competition. Both algorithm 1 and 2 are hill-climbers: they compare today's profits with the previous profits. Implicitly I am assuming that if I were to revert back to the old target I would earn the old profit. This stops being true when competitors are concurrently changing their targets. The maximization algorithm thinks it is maximizing $\Pi(L)$ but it is actually maximizing $\Pi(L_i, L_{-i})$ with no control or knowledge of opponents' workforce L_{-i}. Each agent decision shifts everybody else's profit function. As a setup it is similar to the "Moving Peaks Benchmark" problem Blackwell and Branke (2006) except that peaks are shifted endogenously by each agent rather than by stochastic shocks.

In spite of this I show in the competitive example that the resulting noise is manageable. It stops agents from approaching any steady state, but it does not stop them from approaching equilibrium prices.

6 Zero-Knowledge Firm Examples

6.1 Mathematical Example

In this example I use no software. Prices and quantities are continuous. A Zero-Knowledge firm hires workers from a market with daily labor supply $L = 2w$, it has daily production function $q = L$, and faces the daily demand function $q = 100 - 5p$. The firm is composed of two departments, a HR department hiring workers and a sales department selling goods. The departments act daily, in parallel and independently. The firm maximizes arbitrarily every 10 days using algorithm 1.

For the first 10 days, the target number of workers is 1. Table3 shows the HR PID process.

Table 3: Non-Computational Example of an HR department in a Zero-Knowledge Firm

Day	HR's e_t	HR's $\sum_{i=1}^{t} e_t$	Wages u_t	Workers to Hire y_t^*	Workers Hired' y_t	Daily Production
1	-	-	0	1	0	0
2	1	1	.250	1	5	.5
3	.5	1.5	.325	1	.650	.650
4	.350	1.850	.388	1	.775	.775
5	.225	2.075	.426	1	.853	.853
6	.148	2.223	.452	1	.904	.904
7	.096	2.319	.469	1	.937	.937
8	.063	2.382	.479	1	.959	.959
9	.041	2.423	.487	1	.973	.973
10	.027	2.450	.491	1	.982	.982

At the same time the sales department is using its own PID controller to sell products. The target sales is equal to daily production (which is driven by the HR department) plus leftover inventory. For this example I force initial sale price to be 20.

Table 4: Non-Computational Example of a Sales department in a Zero-Knowledge Firm

Day	Sales's e_t	Sales's $\sum_{i=1}^{t} e_t$	Sale Price u_t	Daily Production	Goods to sell y_t^*	Customers Attracted y_t
1	·	·	20	0	0	0
2	0	0	20	.5	.5	0
3	-.5	-.5	19.875	.650	1.150	.626
4	-.525	-1.025	19.769	.775	1.3	1.156
5	-.144	-1.169	19.759	.853	.996	1.205
6	.208	-.960	19.818	.904	.904	.908
7	.004	-.956	19.809	.937	.937	.955
8	.018	-.938	19.813	.959	.959	.934
9	-.025	-.963	19.806	.973	.998	.970
10	-.029	-.992	19.800	.982	1.011	.999

At the end of Day 10 the maximization algorithm is called and compares profits with 0 workers (which is 0) against the profits with 1 worker target. The firm paid .491 in wages to .982 workers, for a total cost of .482; the firm

produced .982 goods sold at 19.8 a unit for a total revenue of 19.443. The firm's daily profits then are 18.961. Because increasing workers increased the profits (from 0 to 18.961) the maximization algorithm sets the new worker target to be 2. 2 is set as target to the HR department from Day 11, restarting the loop.

The two departments are linked only through production: HR gathers the input, the sales department sells its production. In this particular example, and in the computational examples that follow, HR and production have priority over sales and always happen before. This is not an important assumption, if the sales department acts first the process is identical except that sales actions are delayed by one day (what happened in day 3 will happen in day 4 and so on).

6.2 A Monopolist Example

There is a single firm with two departments: a sales department and an HR department. There is a fixed daily demand for goods as shown in figure 12. The demand is step-wise and discrete. The firm also faces a step-wise discrete supply curve made up of individual workers' reservation wages. The firm must pay a single wage to all employees which explains why the marginal cost curve is steeper than the wage curve (the second worker has reservation wage $16, but hiring him requires raising the first worker wage by $1, hence the marginal cost is $17).

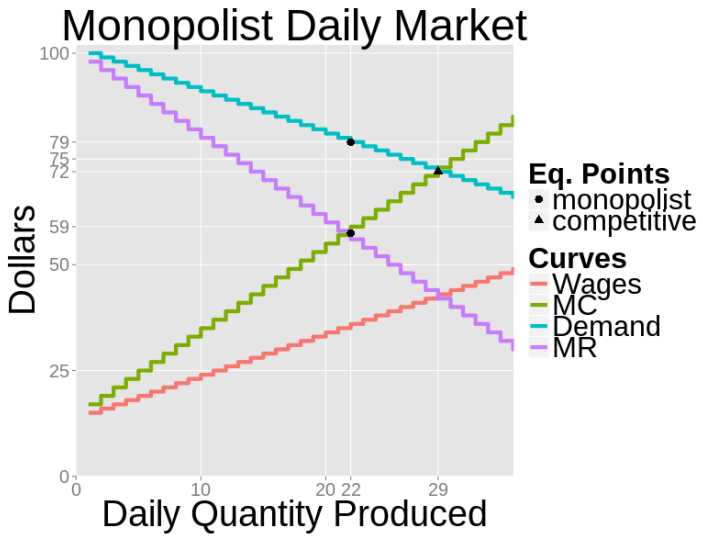

Figure 12: The daily demand faced by the monopolist, the wage curve and the resulting marginal cost curve

Production is constant returns: each worker produces 1 unit of good every

day. There is no capital, no fixed costs and no other inputs. Market is an order book. Everybody places limit orders and crossing orders are automatically filled. The trading price is always the price quoted by the seller. Prices and quantities are always natural numbers. A rational monopolist maximizes profits by hiring 22 workers. The rational monopolist price is $79.

The Zero-Knowledge firm has none of this information. The firm has no knowledge of being a monopolist either. Initially the wage offered is set to 0, the sale price is set to 100. The maximization used is algorithm 1. The maximization wait time is endogenous: 3 weeks after the labor targets have been filled by the HR department.

The firm's daily production and sale price in a sample run are shown in the figures 13 and 14. The Zero-Knowledge firm acts rationally in spite of no knowledge, uncoordinated departments and rudimentary maximization.

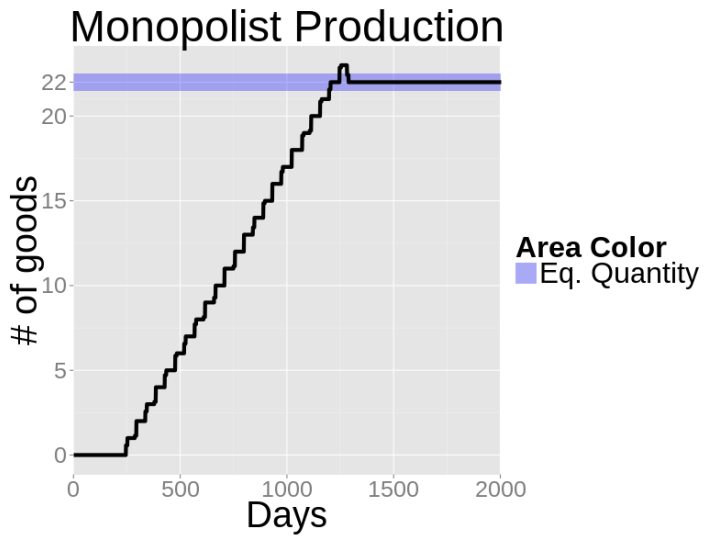

Figure 13: Daily production in a sample run with a single firm

Notice in figure 14 the same temporary undershooting as in the Zero-Knowledge seller example; this undershooting has a different cause: the sales department PID has no foreknowledge of different changes in worker targets. In a way the sales department is continually surprised by changes in production and its PID controller has to catch up. It is the cost of using completely reactive control and total departmental independence.

The results are only slightly different if I use algorithm 2. In this case the firm forever oscillates between hiring 21, 22 and 23 workers ad libitum.

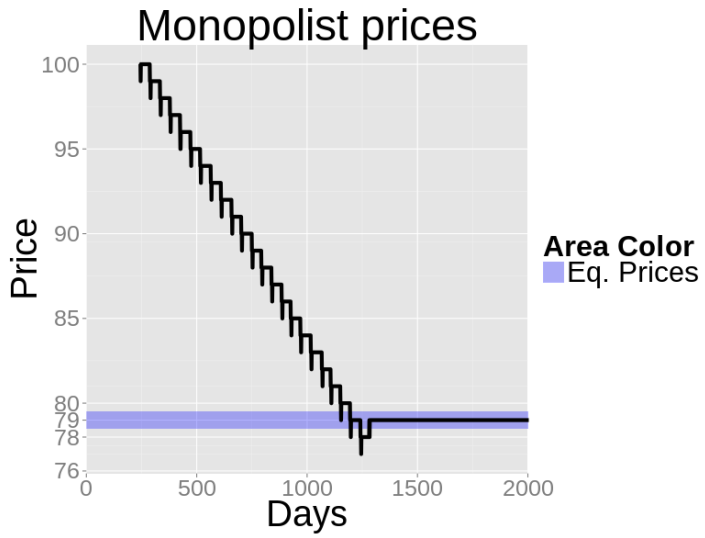

Figure 14: Daily production in a sample run with a single firm

6.3 A Competitive Example

I replicate the market of the section 6.2 and add competition. In this example there are 5 firms in the market. Nothing changes in the internal structure of the firm. The firms have no knowledge of having competitors. Each firm follows algorithm 2 to maximize. The competitive equilibrium price would be $72 and the equilibrium daily production would be 29.

Figure 15 and figure 16 show a sample run. Unlike the monopolist case, the results are more noisy and do not stabilize. Both the quantity traded and the prices orbit around the equilibrium values, but they never settle.

I run the competitive model 5000 times changing only the random seed. I stop each simulation after 5000 days and record final price and quantity. Figure 17 shows the distribution of results. While dispersed, all observations cluster around the market demand function. This shows how with competitive noise, control keeps performing well in keeping production and price linked even when the maximization fails to find the profit maximization quantity. If I focus on prices alone, as in figure 18 I can see that almost all the simulations with competition have prices lower than than the monopoly setup.

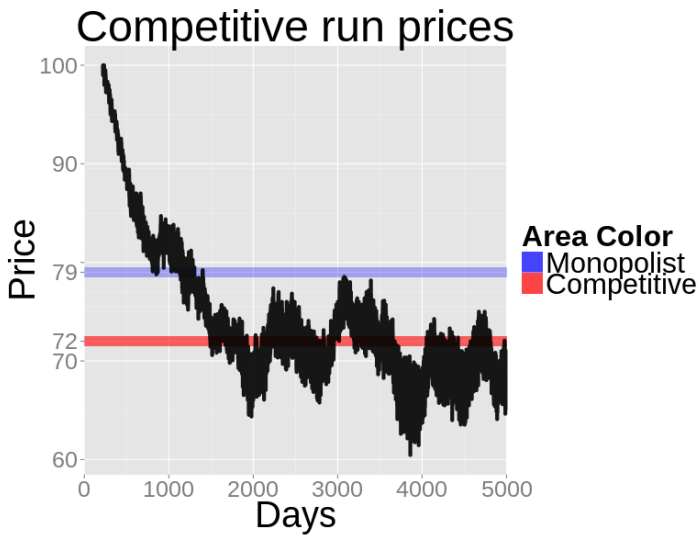

Figure 15: Daily prices in a sample run with 5 firms

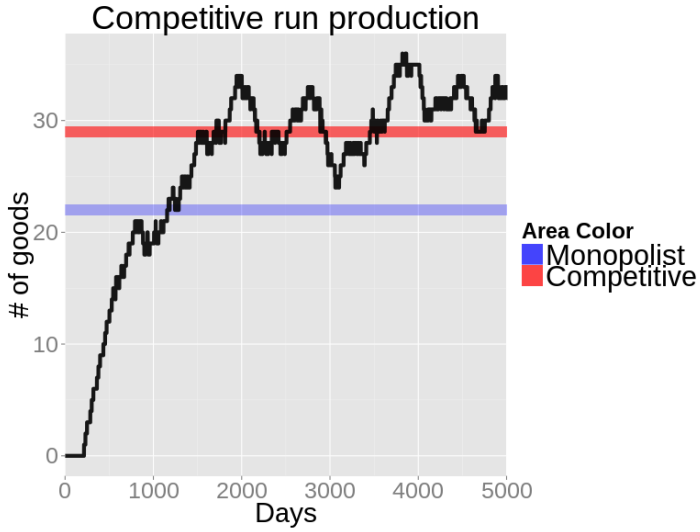

Figure 16: Daily production in a sample run with 5 firms

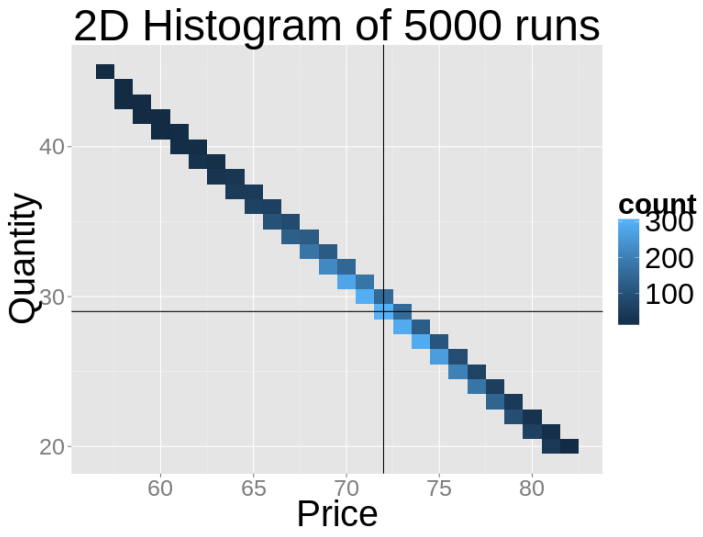

Figure 17: The 2D histogram of price-quantity results of 5000 sample runs of the competitive scenario

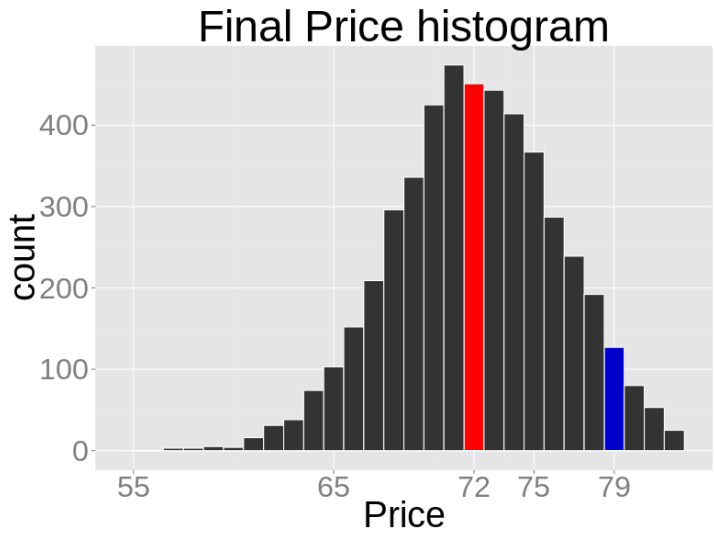

Figure 18: The histogram of prices from 5000 competitive runs. The red bar represents the theoretical competitive prices, the blue bar the theoretical monopoly prices

7 Conclusion

The Zero-Knowledge agents I built solve the monopolist problem in simple markets perfectly but are inaccurate in perfectly competitive markets. This is in part because the hill-climbing algorithms 1 and 2 react to past profits and prices rather than current ones.

There are three assumptions that power the Zero-Knowledge traders that I think should be addressed. First, I have decided that firm's trial and error is done over prices. Thanks to economics surveys by Blinder (1998) and Fabiani, Silvia et al. (2006) I know that price flexibility is uncommon. Prices are more like targets, changing perhaps three times a year.

Second, while Zero-Knowledge firms were created to show how agents can bootstrap correct behavior without looking at prices, there is no reason to assume agents are so autistic. Benchmarking is common-place in any industry. A more realistic model would use more feed-forwarding and, more importantly, more nuanced optimization.

Third, I assumed that demand reacts immediately to changes in prices. This is in line with usual economic assumptions but it has the additional advantage of avoiding the complicated design of controllers that deal with delays between policy changes and results.

In spite of its simplicity, agents with this behavior can provide a simple baseline on which to build other economic agent based models.

References

Axtell, R. (2005). The Complexity of Exchange*. *The Economic Journal*, 115(504):F193–F210.

Bagnall, A. and Toft, I. (2006). Autonomous Adaptive Agents for Single Seller Sealed Bid Auctions. *Autonomous Agents and Multi-Agent Systems*, 12(3):259–292.

Blackwell, T. and Branke, J. (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459 –472.

Blinder, A. S. (1998). *Asking About Prices: A New Approach to Understanding Price Stickiness*. Russell Sage Foundation.

Chen, X. and Deng, X. (2006). Settling the Complexity of Two-Player Nash Equilibrium. In *47th Annual IEEE Symposium on Foundations of Computer Science, 2006. FOCS '06*, pages 261–272.

Cliff, D. (1997). Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments. Technical report.

Cliff, D., Bruten, J., and Road, F. (1997). Zero is Not Enough: On The Lower Limit of Agent Intelligence for Continuous Double Auction Markets. *HP Laboratories Technical Report HPL*.

Currie, M. and Steedman, I. (1990). *Wrestling with Time: Problems in Economic Theory*. University of Michigan Press.

Deng, X., Papadimitriou, C., and Safra, S. (2002). On the Complexity of Equilibria. In *IN PROCEEDINGS OF THE 16TH ANNUAL SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE*, pages 404–413.

Dosi, G., Fagiolo, G., and Roventini, A. (2010). Schumpeter meeting Keynes: A policy-friendly model of endogenous growth and business cycles. *Journal of Economic Dynamics and Control*, 34(9):1748–1767.

Fabiani, Silvia, Druant, Martine, Hernando, Ignacio, Kwapil, Claudia, Landau, Bettina, Loupias, Claire, Martins, Fernando, Matha, Thomas, Sabbatini, Roberto, Stahl, Harald, and Stokman, Ad (2006). What Firms' Surveys Tell Us about Price-Setting Behavior in the Euro Area. *International Journal of Central Banking (IJCB)*.

Foley, D. K. (2010). What's wrong with the fundamental existence and welfare theorems? 75(2):115–131.

Gintis, H. (2007). The Dynamics of General Equilibrium*. *The Economic Journal*, 117(523):1280–1309.

Gjerstad, S. and Dickhaut, J. (1998). Price formation in double auctions. Technical report, CiteSeerX.

Gode, D. K. and Sunder, S. (1993). Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality. *Journal of Political Economy*, 101(1):119–37.

Howitt, P. and Clower, R. (2000). The emergence of economic organization. *Journal of Economic Behavior & Organization*, 41(1):55–84.

Jaffe, W. (1967). Walras' Theory of Tatonnement: A Critique of Recent Interpretations. 75(1):1–19.

Leijonhufvud, A. (1984). Hicks on Time and Money. *Oxford Economic Papers*, 36:26–46. ArticleType: research-article / Issue Title: Supplement: Economic Theory and Hicksian Themes / Full publication date: Nov., 1984 / Copyright © 1984 Oxford University Press.

Mäki, U. (2008). Economics. In Curd, M. and Psilos, S., editors, *The Routledge Companion to Philosophy of Science*, pages 543–554. Routledge, London.

Nowak, A., Rychwalska, A., and Borkowski, W. (2011). Why Simulate? To Develop a Mental Model. *Journal of Artificial Societies and Social Simulation*, 16(3):12.

Ortega, M. and Lin, L. (2004). Control theory applications to the production–inventory problem: a review. *International Journal of Production Research*, 42(11):2303–2322.

Papadimitriou, C. H. (1994). On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532.

Scarf, H. (1960). Some Examples of Global Instability of the Competitive Equilibrium. *International Economic Review*, 1(3):157–172. ArticleType: research-article / Full publication date: Sep., 1960 / Copyright © 1960 Economics Department of the University of Pennsylvania.

Smith, C. A. and Corripio, A. B. (2005). *Principles and Practices of Automatic Process Control*. Wiley, 3 edition.

Åström, K. J. and Hägglund, T. (2006). *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society.