

Dynamic Oligopoly and Monetary Policy: A Deep Reinforcement Learning Approach

Matias Covarrubias

[\[Link to the latest version\]](#)

Abstract

We evaluate the effect of dynamic oligopolistic strategies on the transmission of monetary policy and conclude that concentration increases the effectiveness of monetary policy mainly because fewer firms collude more often, and collusive prices react less to monetary policy. The main innovation of the paper is that we solve the strategic problem of firms by using state-of-the-art artificial intelligence algorithms called Deep Reinforcement Learning. These algorithms learn by experimenting and improving their strategies over time, using only information about their period rewards and the evolution of the state. Additionally, they approximate the policy and value functions using neural nets. This methodology allows us to study market structures with any number of firms, and we generalize the set of strategies to repeated game strategies and non-Markovian strategies. Using this approach, we construct a monetary economy with menu costs and oligopolistic competition at the sector level. After benchmarking with known results in the literature in a game with a finite (but large) horizon, we allow for repeated game strategies by solving the infinite horizon version. We get a range of converging equilibria, from the Markov equilibrium calculated in recent studies to high-markup equilibria that we interpret as tacit collusive equilibria. We find that monetary policy effectiveness, measured by its contribution to the volatility of log consumption, is up to 40% larger with collusive strategies versus non-collusive strategies. By inspecting the converging policies, we observe that collusion is supported by strong reactions to the price of the competition that ensure that deviations are costly. This stronger price complementarity accounts for the increased effectiveness. Then, we vary the number of firms per sector and show that fewer firms imply more effective monetary policy, mainly because collusion is easier.¹

¹I am grateful to my advisors Ricardo Lagos, Thomas Philippon, Stijn Van Nieuwerburgh and Douglas Gale for their support. I would also like to thank Jiacheng Li for his excellent research assistance with the methodological appendix and Agustin Covarrubias for his essential contribution to the server infrastructure and the Python platform. I benefited from discussions with Simon Mongey, Jaroslav Borovicka, Diego Perez, Virgiliu Midrigan, Jess Benhabib, Mark Gertler and Ennio Stachetti. All the remaining errors are mine.

1 Introduction

What is the effect of dynamic oligopolistic strategies on monetary policy effectiveness? In [Covarrubias, Gutiérrez, and Philippon \[2020\]](#), we review the evidence on concentration trends in U.S. industries and discuss the different hypotheses behind the sharp increase in concentration in the last decades. These trends have raised the natural question of what are the implications of higher concentration for the transmission of monetary policy [[Mongey, 2021](#), [Wang and Werning, 2020](#)]. If the typical market has few firms competing, we want to understand how their strategic behavior affects price setting dynamics. The main challenge with answering this question is that solving for optimal strategies in a macroeconomic environment with realistic idiosyncratic and monetary shocks is computationally expensive. For example, [Mongey \[2021\]](#) declares the problem to be intractable with more than two firms per sector using traditional computational methods which attempt to find a fixed point in policies directly. Furthermore, given the existence of multiple equilibria in dynamic pricing games [[Maskin and Tirole, 1988](#)], quantitative applications focus on a subset of optimal Markovian policies obtained by iterating backward from a finite horizon game.

In this paper, we generalize the set of strategies and increase the number of firms per sector by assuming that firms learn their optimal policies from experience using state-of-the-art artificial intelligence algorithms called Deep Reinforcement Learning [[Schulman et al., 2017](#)]. A key aspect of this learning process is exploration: during the learning phase, firms have stochastic policies, so they do not always make optimal price adjustment decisions. After firms feed their actions to the system, they observe the reward they got that period (i.e. their profit) and the evolution of the state. With this information, they update the parameters of their policy, which is represented as a neural net mapping states to actions. As learning progresses, the policy becomes less stochastic and eventually converges to a specific strategy. As part of the algorithm to obtain the optimal policy, firms also estimate a neural net that outputs the value of taking a particular action in a given state. This learning mechanism is forward-looking, and as a validation, we show that it converges to the optimal policy in a Real Business Cycle model after roughly 50 updates to the neural nets². The two advantages of this methodology are that it is scalable to many firms and provides an equilibrium selection mechanism: we get to observe the range of equilibria that are learnable by sophisticated algorithms that experiment with different strategies. Furthermore, the methodology is very general, and modifying the economy under study requires only modifying the game.

Our main result is that tacit collusive strategies increase monetary policy effectiveness by up to 40%. Additionally, market concentration has a significant and positive effect on monetary policy effectiveness, mainly because collusion is harder for more than two firms. We arrive at these conclusions by studying a rich quantitative multi-sector economy with oligopolistic competition at the sector level and nominal rigidities in the form of menu costs. Each firm receives idiosyncratic shocks in addition to an aggregate monetary shock. In the initial calibration, the economy matches data on average markup, frequency of price adjustment, and size of adjustment for the U.S. between 1985 and 2016.

We perform the following exercises. First, we validate the methodology by comparing our results with

²See Appendix A for methodological details on this exercise and on the specific algorithm we are using.

the results obtained with traditional dynamic programming methods in a tractable specific case. In particular, the framework we study nests the economy presented in [Mongey \[2021\]](#)'s, which has 2 firms per sector, restricts strategies to be Markovian (so there is no memory), and selects a particular Markovian equilibrium by solving the problem backward from a finite horizon game. We use this specific case as a benchmark by solving a finite (but large) horizon version which prevents repeated games strategies. We show that the statistics on markups and price adjustment in the finite horizon version match [Mongey \[2021\]](#)'s, and the policies we obtain follow a set of expected behaviors.

Then, we generalize the set of equilibria under study by solving the infinite horizon version and allowing firms to have limited memory. In this version, we observe higher markups and more dispersion in the markups agents converge to. When analyzing the distribution of equilibria, we see that a large proportion of the experiments converge to higher markup equilibria which we interpret as tacit collusive equilibria. We compare monetary policy effectiveness in the finite-horizon equilibrium with low average markup versus a representative infinite horizon equilibrium with high average markup and conclude that tacit collusive strategies observed in the high markup equilibrium increase monetary policy effectiveness. When inspecting the policies, we find stronger price reactions to changes in the price of the competition in collusive equilibria, that is, stronger price complementarities. Under these complementarities, prices are determined mostly by the price of competition, so there is a large degree of "pricing-to-market" [[Krugman, 1986](#)], and they react less to monetary shocks. In a model with dynamic pricing, complementarities are not an off-equilibrium phenomenon, but can be observed by looking at a firm's response to a change in the markup of the competition in the previous period. With collusion, those complementarities increase because collusive strategies are supported by strong reactions to changes in the markup of the competition.

Finally, we increase the number of firms in each sector, and evaluate the effect of concentration on monetary policy effectiveness. In the non-collusive equilibrium, we confirm [Wang and Werning \[2020\]](#)'s result that under empirically reasonable parameters, more concentrated markets lead to mildly more effective monetary policy since pricing of large firms is more responsive to the competition and less responsive to fundamentals. In the collusive equilibrium, there is an additional increase in monetary policy effectiveness due to more concentrated markets leading to more collusion. For example, going from 3 to 2 firms increase effectiveness only by 5% in the non-collusive case, versus 25% for the collusive case.

Methodologically, the main innovation of this paper is the use of modern artificial intelligence algorithms that employ neural nets and sophisticated experimental architectures designed to obtain optimal strategies in high-dimensional problems³. While there are precedents of using traditional RL algorithms to solve challenging dynamic games [[Fershtman and Pakes, 2012](#)], this paper is the first to implement modern deep RL learning architectures to general equilibrium problems. This algorithmic upgrade is better suited to macroeconomic applications that study fully fledged economies with multiple sectors and in which actions are better represented as continuous variables. For example, when we solve the problem with 5 firms per

³[Chen et al. \[2021\]](#) also apply Deep RL to Macroeconomic problems. The main difference is that they focus only on representative agent problems, while here we focus on multi-agent strategic problems. Their comparison of Deep RL traditional approaches to adaptive learning is complementary to my paper. In particular, they show that Deep RL learns the rational expectation equilibria in edge cases where adaptive learning fails.

sector in our framework, we have 7 state variables and 2 continuous actions per firm. Solving this would be unfeasible for traditional RL algorithms that use tables to store value function estimates for all action-state pairs.

Related Literature. In macroeconomics, two approaches have been used to study dynamic oligopolistic settings in general equilibrium⁴. [Mongey \[2021\]](#) uses computational methods to solve for optimal Markovian strategies in a model with menu costs, duopoly at the sector level and stochastic monetary and idiosyncratic shocks. His main result is that, compared to monopolistic competition models, duopolistic strategies increase the effectiveness of monetary policy, since strategic behavior make the prices of firms strategic complements. In particular, when an aggregate shock hits a firm, the fact that the other firm also received the shock alleviates the need to adjust, which implies lower pass-through and higher effectiveness of monetary policy. Alternatively, [Wang and Werning \[2020\]](#) use continuous time methods to analytically characterize the Markov equilibria of a stationary economy with Calvo pricing and general n firms per sector competing strategically. In this setting, the authors solve for the response of aggregate consumption to a permanent monetary policy shock and conclude that the steeper the reaction function of a firm, the lower the pass-through of its price to a monetary policy shock. Then, they show that under empirically plausible preference parameters, fewer firms imply steeper reaction functions, so higher concentration leads to more effective monetary policy. Both results highlight a similar mechanism: strategic behavior introduces dynamic complementarities in pricing, which means that the impact on incentives to adjust is lower when the shock affects both firms.

Compared to these two approaches, the methodology employed in this paper expands on the set of strategies that can be analyzed quantitatively in a model with full idiosyncratic and aggregate shocks. In particular, with respect to [Mongey \[2021\]](#), we relax the assumption of Markovian strategies and finite horizon⁵. By doing so, we can analyze strategies that arise in repeated games, such as collusive strategies that punish deviations. Also, the deep RL methodology implemented here provides increased tractability which permits us to study frameworks with an arbitrary number of firms per sector without sacrificing the richness of the stochastic processes. This increased tractability comes as a by-product of the model-free simulation approach and the neural net architecture.

Our approach to the study of multiple equilibria and collusive strategies is also informed by the recent literature on algorithmic collusion [[Calvano, Calzolari, Denicolo, and Pastorello, 2020](#), [Asker, Fershtman, and Pakes, 2021](#)]. The topic of tacit collusion between pricing algorithms has become more relevant due to the increasing share of prices in the economy that are fixed automatically using machine learning. My paper builds on these papers and shares their approach of studying the range of observed multiple equilibria by solving many learning experiments and looking at the resulting distribution of converging strategies. Besides

⁴[Jun and Vives \[2004\]](#) also study dynamic oligopolistic pricing in a single market from a theoretical perspective. [Rotemberg and Woodford \[1992\]](#) study repeated game strategies in a general equilibrium model with no nominal rigidities.

⁵This assumption is not explicit, but rather a consequence of the solution method. [Mongey \[2021\]](#)'s main algorithm starts from a static Nash conjecture and solves the bellman equations backwards, which breaks down the sub-game perfect equilibrium with repeated game strategies.

studying a more complex dynamic game and using deep RL algorithms, an important difference of our paper is that we are using RL as a computational methodology to obtain optimal strategies by (potentially human) agents. We are not solely focusing on the consequences of literal algorithmic pricing.

My paper is also related to recent contributions to computational macroeconomics that emphasize the power of using neural nets as function approximators for the policy. These contributions can be organized around three key implementation choices, namely the use of simulation-based learning, discrete but large number of agents, and exploitation of symmetric policy mapping with heterogeneous state realizations. [Maliar et al. \[2021\]](#) use deep learning to solve a "discrete agents" version of [Krusell and Smith \[1998\]](#). Their method is simulation-based, which means that the iteration step consists of simulating the path of the economy under the current policy, calculating a loss function from that experience, and then updating the policy in a direction that minimizes such loss. [Azinovic et al. \[2019\]](#) use a similar algorithm, which shares the use of neural nets and the simulation-based approach, but specialize the loss function to focus only on Euler errors. Alternatively, [Kahou et al. \[2021\]](#) present a deep learning approach in which the focus is on the fact that policy maps are symmetric among agents, despite their states being heterogeneous, so we can use a single neural net to represent their policy. Besides the use of neural nets, my paper shares the simulation-based approach, the focus on many discrete agents, and the exploitation of symmetry in the policy mapping. However, regardless of these similarities, the deep RL approach differs significantly, since with model-free algorithms agents do not minimize Euler errors or related model-based equations. Instead, the agents experiment and update the policy and value functions by observing their rewards and the evolution of the state given the actions they took. As such, they do not have embedded in them the traditional assumptions that are imposed when taking first order conditions, such as assuming that agents take aggregates and prices as given. This agnostic approach to optimization makes deep RL more suitable for the analysis of dynamic strategic behavior, while the deep learning approaches we have discussed are preferable for the calculation of recursive competitive equilibria, in which we make use of first-order conditions to impose abstractions that facilitate the interpretation of the results.

Finally, the framework we study has been developed gradually through a rich literature that attempts to reconcile observed levels of monetary non-neutrality with the micro evidence on price adjustments. The main puzzle is that on the one hand the estimated slope of the Phillips curve is low [[Nakamura and Steinsson, 2010](#), [Hazell et al., 2020](#)], which is consistent with highly effective monetary policy. On the other hand, models that match the frequency and size of price adjustment imply very low real effects of monetary policy shocks. The first paper to make this point was [Golosov and Lucas \[2007\]](#), which contrasted their results with the evidence presented on [Bils and Klenow \[2004\]](#) and concluded that the data on price adjustment was consistent with monetary neutrality in a model with menu costs. Since then, a number of papers have tried to alleviate the mechanism behind this neutrality result, generally through different sources of pricing complementarities [[Burstein and Hellwig, 2007](#), [Nakamura and Steinsson, 2010](#)] or the introduction of small price changes through multi-product firms or random menu costs [[Midrigan, 2011](#), [Alvarez and Lippi, 2014](#)]. My paper uses as a benchmark the most recent of such efforts, [Mongey \[2021\]](#), which argues that strategic complementarities are especially adept at generating monetary non-neutrality without resorting to

unrealistically large menu costs, since the mechanism only applies to the response to aggregate shocks and not to idiosyncratic shocks. In this context, this paper suggests that tacit collusive strategic behavior in concentrated markets can help explain why monetary policy is so effective. It is worth noticing that even though we use many of the modeling techniques developed in this literature, the question we are pursuing here is different. We are interested in analyzing how pricing behavior changes when we look at different equilibrium strategies or market structures. We do not propose a different model that can provide efficient monetary while targeting fixed price adjustment statistics. Instead, we start from a model that hits those statistics, and then ask how they change under a number of counterfactuals.

Preview of the framework. There is a large number of sectors and, in each sector, N^f firms that choose markups subject to idiosyncratic cost shocks and aggregate monetary shocks that shift the markup in the case of no adjustment, which we call "no-reset" markups. In each period, firms observe the "no-reset" markups of firms in their sector, in addition to last periods' aggregate markup and money growth. Based on this information, they decide the threshold menu cost under which they adjust and the reset markup in case they do adjust. As we will show in Section 3, this game can be derived from a fully characterized dynamic model with a representative household à la [Goloso and Lucas \[2007\]](#) and with firms that face idiosyncratic shocks to both marginal cost and preferences for the product⁶. The resulting markups game is stationary and the role of monetary policy is easy to interpret. Crucially, the direct effect of monetary policy in this game is simply to reduce the markups of all firms, so the question on its efficiency is equivalent to the question on the level of pass-through to an aggregate cost shock.

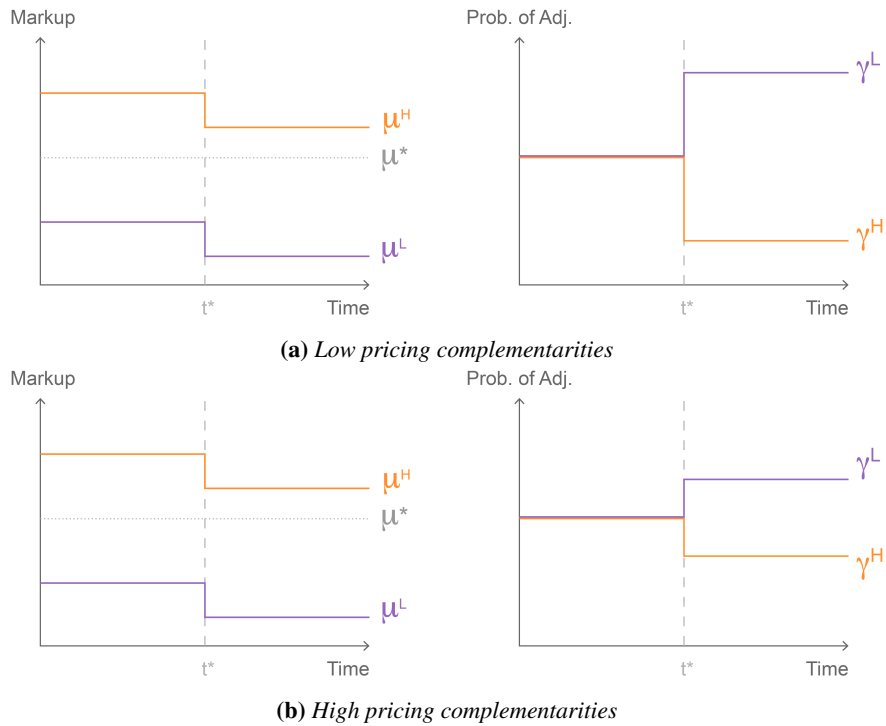
The implementation of this game as an RL learning environment is as follows. First, we initialize the game with some random level of markup for all firms and initial realization of the shocks. After observing the initial observation, firms will submit their threshold menu cost and reset markup to the environment based on their current policy function. Given the adjustment decisions of all firms, the environment calculates each firm's profits, updates the stochastic processes, and exposes the new observations to the firms. To improve strategies, firms will experiment and learn from experience. In particular, every 20 years of monthly experience, all the observed transitions are collected and fed to the gradient descent algorithm to update the policy and value function parameters. After 5000 of such updates, we verify whether behavior has settled and analyze the resulting policy functions. For each version of the game, we run 50 parallel learning trials, so we can observe the distribution of convergent behavior. This exercise follows the recent literature on algorithmic collusion [[Calvano et al., 2020](#), [Asker et al., 2021](#)], which also studies a distribution of outcomes in repeated games.

Mechanism. The main argument of the paper is that collusive strategies affect monetary policy effectiveness through the steepness of the reaction function of firms. The argument for why collusive strategies imply steep reaction functions is straightforward: steep reaction functions imply higher costs of deviating

⁶This kind of shocks are referred to as "quality shocks", in the sense that they affect both the cost of production and the utility that consumers get from it.

from a markup equilibrium, so they serve as a punishment device. But why do steeper reaction functions increase monetary policy effectiveness? In the framework we are going to study, monetary policy shocks translate into a reduction in the markup of all firms. Given nominal rigidities, in a typical market, there is a firm whose markup is above their desired markup, the high-markup firm, and there is a firm whose markup is below its desired markup, the low-markup firm. In the left panels of Figure 1, we see the effect of a positive monetary shock at time t^* in the markups of these two types of firms. For a high-markup firm, the monetary shock is good news, since now the markup is closer to the optimal markup μ^* . Thus, we expect the incentive to adjust for the high-markup firm to be reduced after the shock. In contrast, for a low-markup firm, the shock pushes its markup farther away from μ^* , so we expect her incentive to adjust to increase after the shock. In the right panel of the figure see how the probability of adjusting changes for each type of firm. We distinguish between two cases. In top panels (a), we present the benchmark of low or no pricing complementarities. In that case, firms are unaffected by the fact that the other firm also received the shock, so its probabilities of adjustment follow our predicted changes in incentives to adjust.

Figure 1: *The effect of a monetary shock on markups and probability of adjustment*



Note: This figure is a simplification. It shows the effect of a monetary shock on markups and probability of adjustment. Within each graph, we distinguish between a high-markup firms, which are above their optimal reset markup, and low-markup firms. In panel (a), we show the case of low pricing complementarities, and in panel (b) we show the case of high pricing complementarities. The main simplification is that reset optimal markups are not necessarily the same for both types of firms, and they are affected by the monetary shock as well. Nevertheless, the key mechanism is better understood by analyzing incentives to adjust.

Panel (b) shows the case when firms have high levels of pricing complementarities. Compared to panel (a), there is an additional effect that moderates the reaction of the probabilities of adjustment. In addition

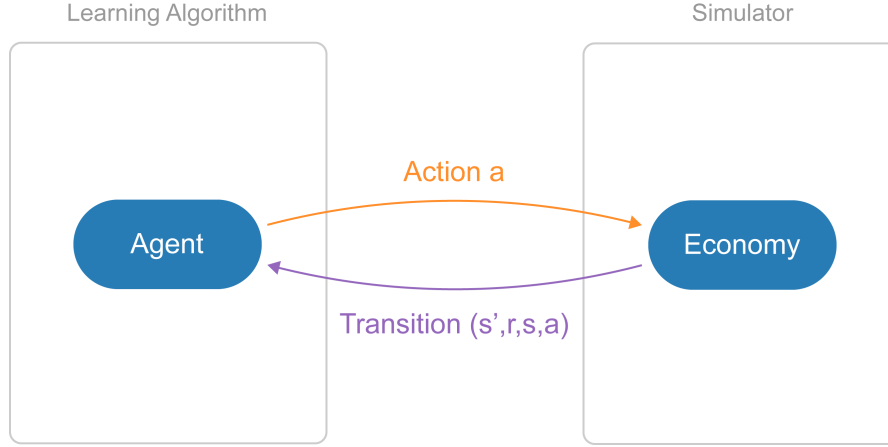
to the direct effect of the monetary shock on the distance of current markup to optimal markup, each firm considers the fact that the other firm markup also changes. In particular, for a low-markup firm, two things have changed after the shock. First, the new markup is farther from the optimal. Second, the profit level in the case that she adjusts to the optimal μ^* has decreased since the high-markup firm now has a lower markup. An analogous argument applies to the high-markup firm: the new markup is closer to the optimal, but the profits at the optimal markup increase since the other firm does not have such a low markup. This second effect reduces the impact of monetary policy on the incentives to adjust for both types of firms. Since the impact of monetary policy on price adjustment is lower, we get a stronger real effect of monetary policy.

Outline. In Section 2 we introduce the Deep RL methodology and show its performance on a Real Business Cycle Model. In Section 3 we present the model that we are going to study. In Section 4 we discuss the results. Finally, in Section 5 we conclude.

2 A Primer on Reinforcement Learning

From the perspective of computational economics, RL algorithms can be understood as adaptations of dynamic programming techniques to the problem of online learning. As illustrated in Figure 2, the algorithm improves the policy function by interacting with the system it’s trying to control, and observing the period rewards (e.g., the period utility) and the evolution of the state. A key element of this process is experimentation. The algorithm does not always choose the policy that it currently believes to be optimal, but to some degree it will try out actions and store the perceived consequences. A major advantage of such an approach is that learning does not require a mathematical model of the system; the structure of the problem is discovered through experience. This feature explains why technological advances in the algorithms have been developed in the context of solving robotic tasks [Lillicrap et al., 2015] and strategic games such as chess and go [Mnih et al., 2013, 2015, Silver et al., 2017]; these problems do not have a known mathematical structure that can be leveraged to apply standard optimization techniques.

Figure 2: Online Learning Problem



Note: This figure illustrates the online learning process. An agent (i.e. firm), submit actions (i.e. saving rate) and the environment calculates the evolution of the state and exposes the transition tuple to the agent. The transition tuple is $\{r, s', s, a\}$, where r represents reward of the agent in that period, s' represents new state, s represents the old state and a represents the action. The agent can use the tuple to improve their policy in a forward-looking manner.

Besides its applicability to unstructured problems, this methodology can be used to solve structured problems in which leveraging the mathematics of the system is challenging or unfeasible. Two examples in economics are problems with multiple-equilibria and high-dimensional problems. For problems with multiple-equilibria, RL can be used as an experimental setting to study the set of solutions learned through recurrent experience. To my knowledge, the first paper that used such an approach was [Marimon, McGrattan, and Sargent \[1990\]](#), who used pre-RL artificial intelligence algorithms to study equilibrium selection in the canonical [Kiyotaki and Wright \[1989\]](#) monetary framework. More recently, the literature on algorithmic collusion [[Asker et al., 2021](#), [Calvano et al., 2020](#)] follows a similar paradigm in the context of infinitely repeated strategic games, with the distinction that they are using RL as a behavioral description of pricing algorithms, not of human agents.

For high-dimensional problems, we can use RL to bypass the explicit solving of a fixed point between expectations and policy function in each point of the state space's grid. Agents will learn optimal behavior in the parts of the space that are visited more often, and the expectations will be constructed through experimentation. This feature of RL has not been exploited in economics yet. An important roadblock to implementing RL as a computational method is that with traditional methods, we can corroborate that the policies are a solution to the system of differential equations describing the equilibrium. In contrast, in RL we can only inspect the resulting policy and value functions and evaluate its performance. In light of this, to show that state-of-the-art RL algorithms can indeed get close to the exact solution, we will use benchmarks with known solutions.

Next, we will review the core of the algorithms and discuss the theoretical literature on their convergence properties. We will properly define the game as a mathematical object and distinguish between single-agent problems, where only one policy controls the system, and multi-agent problems. From a learning

perspective, the key difference between both types of problems is that the games are not stationary in multi-agent problems. This is because agents learn and experiment at the same time other agents are learning and experimenting.

2.1 Single-agent Problems

The development of RL has focused on the solution of control problems that are Markovian. In particular, the games are formalized as a stationary Markov Decision Process (*MDP*). In each period t the agent observes a state s_t from a state space S and then selects action a_t from an action space A . After acting, the agent will receive a period reward r_t and observe the next state s_{t+1} , which is drawn from a probability distribution $p(\pi_t, s_{t+1}|s_t, a_t)$. To finalize the description of the *MDP*, we need a probability function for the initial state $\rho(s_0)$. Hence, an MDP can be defined as a tuple $\{p(\pi_t, s_{t+1}|s_t, a_t), \rho(s_0), S, A\}$, where most of the structure of the game is encoded in the transition $p(\pi_t, s_{t+1}|s_t, a_t)$. While the state and action spaces may be infinite in principle, the first RL algorithms were developed to act on games with discrete and finite action and state space, such as chess.

Q-Learning. The q-learning algorithm was first presented in [Watkins \[1989\]](#), and it represents the backbone of the RL literature. The optimization problem is:

$$\max_{a \in A} \sum_{t=s}^{\infty} \beta^{s-t} E_t[r_{t+s}]$$

We can define two value functions. First, we have the state-value function, defined as the expected discounted rewards given that you are in a current state s :

$$V(s) = \max_{a \in A} \{E[r|s, a] + \beta E[V(s')|s, a]\}$$

We can also define the action-value function, which defines the value of being in a state and takes a particular action:

$$Q(s, a) = E[r|s, a] + \beta E[\max_{a' \in A} Q(s', a')|s, a] \quad (1)$$

We initialize the algorithm with an arbitration action-value function $Q_0(s, a)$. From there, the learning loop is the following:

1. **Act:** With ϵ probability, explore. If not explore, agent chooses $a = \arg \max Q(s, a)$ and observes $\{r, s', s, a\}$.

2. **Learn:** Use the tuple $\{r, s', s, a\}$ to update the current Q function. Specifically, the new estimate of Q is going to be a weighted average between the old Q and the realization of the right side of the Q -value function:

- Old Q : $Q(s, a)$.
- Target Q : $r_t + \beta \max_{a' \in A} Q(s', a')$.
- New Q :

$$Q'(s, a) = (1 - \alpha)Q(s, a) + \alpha[r + \beta \max_{a' \in A} Q(s', a')]$$

where α is the learning rate, a key parameter in the algorithm.

The crucial insight of the algorithm is that we can use an observed transition $\{r, s', s, a\}$ to recover a realization of the right side of the action-value function in Equation 1, $E(r|s, a) + \beta E[\max_{a' \in A} Q(s', a')|s, a]$. Then, exploration allows us to get enough realizations to recover the full expectation. In this sense, the learning rate α guarantees that we are not putting too much weight on any given realization.

As an example, let's look at how this learning loop would work in a real business cycle model. A representative household chooses how much to consume and how much to invest to maximize $\sum_{t=0}^{\infty} \beta^t u(c_t)$. The agent produces $y_t = z_t k_t^\alpha$. Productivity z_t follows a two-state Markov process with transition probability $\mathcal{P}$. The game can be represented as reinforcement learning game as follows:

- *Environment Reset:* The game starts with random initial states around the steady state k^{ss} : $k_0 \sim U[0.2k^{ss}, 1.2k^{ss}]$ and z_0 is a randomly assigned to *low* or *high*.
- The learning loop is:
 1. *Agent Act:* For $t \geq 0$: Given an observation of state $\{k_t, z_t\}$, the agent chooses a random saving rate $s_t \in [0, 1]$ according to stochastic policy $a_t(k_t, z_t)$.
 2. *Environment Step:* Given s_t , the environment calculates $c_t = (1 - s_t)y_t$, updates the states $z_{t+1} \sim \mathcal{P}(z_{t+1}|z_t)$; $k_{t+1} = (1 - \delta)k_t + s_t y_t$, and exposes to the agent:
 - Reward: $u(c_t)$
 - Next State: $\{k_{t+1}, z_{t+1}\}$
 3. *Learn:* After observing one (or many) transitions $\{u_t, \{k_{t+1}, z_{t+1}\}, \{k_t, z_t\}, s_t\}$, the agent updates the parameters of its policy $a(S)$ using RL.

In terms of the theoretical properties of the algorithm, we can obtain guarantees of convergence to the exact solution for single agent problems with finite actions and state spaces. In particular, [Tsitsiklis \[1994\]](#) performs an extensive convergence analysis of q-learning algorithms. Given a finite (discrete) state space and action space, Q-learning converges globally insofar as all state-action pairs are visited infinitely often. As in convergence proofs for value function iteration, the proof of convergence in [Tsitsiklis \[1994\]](#) is based on the contraction mapping theorem and Banach fixed point theorems.

2.2 Deep Reinforcement Learning

The Q-learning algorithm pioneered a literature in reinforcement learning that until recently represented a mostly academic subfield in computer science. The key result that convinced a wider audience of its applicability was published by [Mnih et al. \[2013\]](#), which showed that using neural nets as function approximators, the same algorithm could learn to perform well in several Atari games from scratch. While the idea of using function approximators for the Q-function was not novel, the fact that the same algorithm could learn to play a diverse set of games opened the gates for experimentation in many fields. At its core, the algorithm is a version of Q-learning, wherein the table storing the Q values for each action and state is replaced with a parametric function $\hat{Q}(s, a) \equiv \mathcal{N}_\rho \sim Q(s, a)$. The fact that neural nets were employed as approximators meant that the tools developed for supervised machine learning, such as gradient descent and tensor libraries, could be used to train RL algorithms.

More specifically, the algorithms consist of using batches of observed transitions $\{r_t, s_{t+1}, s_t, a_t\}$ to improve the estimation of parameters ρ of the Neural Net. The improvement is minimizing the following loss function:

$$\mathcal{L}(\rho) = \sum_{t=1}^N \left[\left(\hat{Q}(s_t, a_t) - (r_t + \beta \max_{a_{t+1} \in A} \hat{Q}(s_{t+1}, a_{t+1})) \right)^2 \right]$$

in observed transitions $\{r_t, s_{t+1}, s_t, a_t\}$. Notice that, in the loss functions, we have the elements of the transition tuple and the parametrized Q values. With that, we can calculate how a marginal change in the parameters of the Q -values would affect the loss function, which gives the necessary gradient to perform gradient descent.

The most common Neural Net, called a Fully Connected Network, is a composite function made of nested linear functions:

$$\mathcal{N}_\rho(x) = \sigma_K(W_K \dots \sigma_2(W_2 \sigma_1(W_1 x + b_1) + b_2) \dots + b_K)$$

where x is the input (in our case a vector with state-action pairs (s, a)), $W_i \in \mathbb{R}^{m_{i+1} \times m_i}$ are the weights, $b_i \in \mathbb{R}^{m_{i+1}}$ are the biases, and $\sigma(\cdot)$ are nonlinear functions. Each layer $W_i x + b_i$ can be thought of as taking the input vector and running several linear regressions in parallel⁷. After the input has gone through one layer

⁷The number of regressions in each layer is the number of rows in the weight W_i , also referred to as the number of nodes in

of regressions, a nonlinear function is applied to the output of each regression. Given this nonlinear layer, by moving the bias in the regression you can control how much nonlinearity you have. After the nonlinearity is applied, the next layer takes the output of each regression and creates a new stack of regressions.

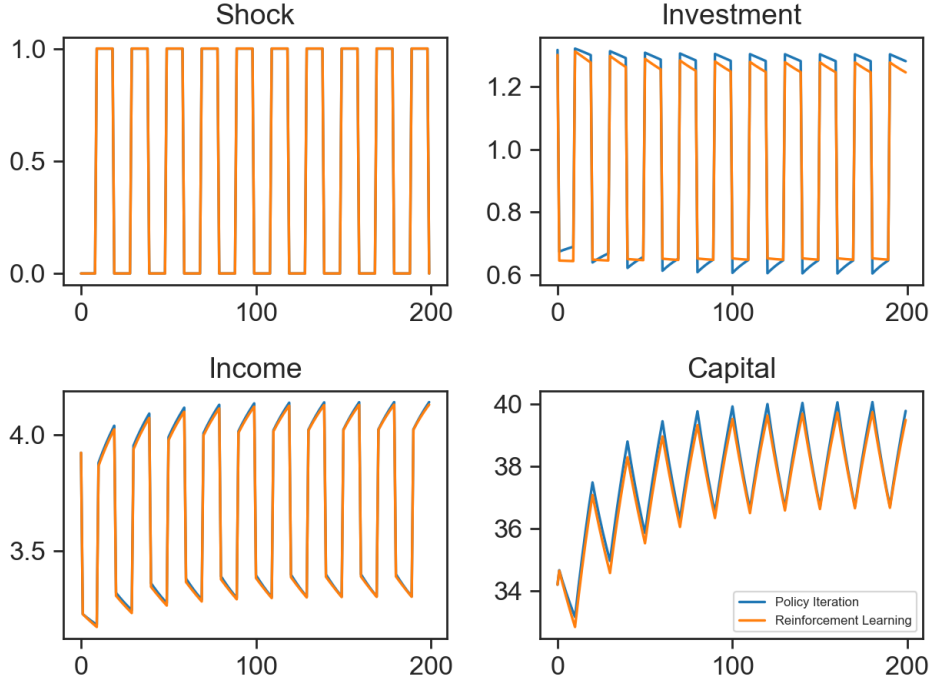
Besides being a universal approximator, a low bar among function approximators, Neural Nets are suitable for high dimensional and nonlinear applications for three reasons. First, the number of parameters grows arithmetically when we increase the dimensionality of the input. In contrast, common approximators have parameter counts that grow exponentially [Grohs et al., 2019]. Second, Neural Nets are well suited for parallel computation. In particular, the estimation of neural nets makes heavy use of matrix multiplication, one of the most parallelizable operations. Third, their layered structure makes them easy to manipulate to accommodate different input and output types. For example, if your output has the form of probabilities for a number of categories (as in categorization problems), you just need to replace the last layer with a softmax function⁸.

As an illustrative benchmark of the performance of deep RL algorithms, in Figure 3 we see a comparison of the policy attained by deep RL versus the solution calculated using the GDSGE [Cao et al., 2020] policy iteration framework with a fine grid of 101 points. The methodological details are in Appendix A. After 50 updates to the neural net (roughly 1 minute of computation, the RL-trained policy achieved a discounted reward of 97.594, compared to 90.598 attained by the policy iteration solution.

that layer.

⁸There are other more technical benefits such as higher robustness to local versus global minima and absence of issues with basis-aligned approximators Wang and Werning [2020].

Figure 3: *Reinforcement Learning vs Policy Iteration in an RBC model.*



Note: This figure shows the performance of two algorithms in solving a Real Business Cycle Model with two productivity states. The orange line is the reinforcement learning solution and the blue line is the policy iteration solution. Given that we used a fine grid, the policy iteration solution can be interpreted as the exact solution. The upper left panel shows the shock. The upper right panel shows investment for both algorithms. The bottom left panel shows the income process and the bottom right panel shows capital level. The Policy Iteration solution was calculated using the GDSGE policy iteration framework. See Appendix A for methodological details.

To measure how well our training outcome compares to the real optimal solution achieved by the GDSGE method, we simulated an alternative path based on random selection of policies. This random policy achieved a discounted reward of 86.32 over 400 periods. Hence, the improvement brought by the trained policy as a fraction of the improvement given by the real optimal policy is $(90.41 - 86.32) / (90.44 - 86.32) = 99.99\%$.

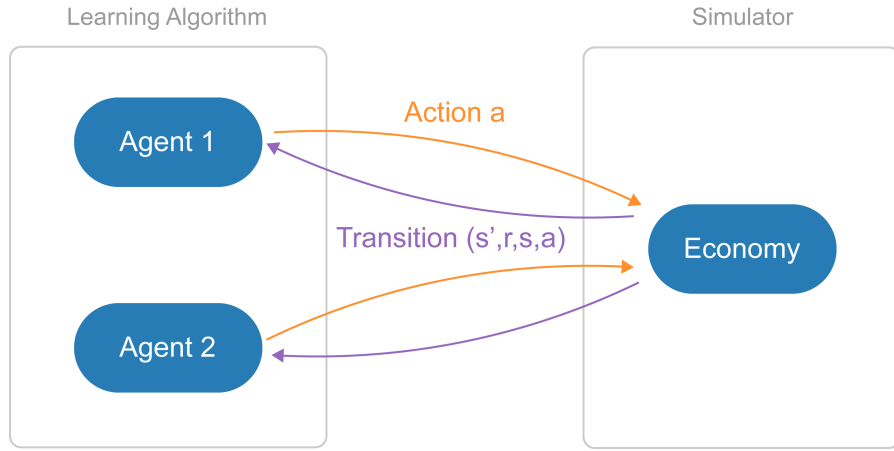
2.3 Multi-Agent Reinforcement Learning

In multi-agent games, there are several agents acting and learning simultaneously (see Figure 4). This requires us to use a different formalization, as Markov decision problems are specific to single agent problems. In particular, we formalize an economy as a Partially Observed Markov Game. In each period:

- n agents indexed by i choose an action $a_i \in \mathcal{A}_i$, after observing a state $s_i \in \mathcal{S}_i$.
- Let $\mathcal{S}$ be the set of states, that is, the collection of all possible states S_i .

- The initial states are determined by a distribution $\rho : \mathcal{S} \mapsto [0, 1]$.
- To choose actions, each agent i uses a stochastic policy $\pi_{\theta_i} : \mathcal{S}_i \times \mathcal{A}_i \mapsto [0, 1]$.
- The combined actions produce the next state according to the transition function $p : \mathcal{S} \times \mathcal{A}_i \times \dots \times \mathcal{A}_N \mapsto \mathcal{S}$.
- Each agent i observes its reward as a function of the state and agent's action $r_i : \mathcal{S} \times \mathcal{A}_i \times \dots \times \mathcal{A}_N \mapsto \mathcal{R}$, and receives a private observation $s'_i : \mathcal{S} \mapsto \mathcal{S}_i$.

Figure 4: Online Learning Problem



Note: This figure illustrates the online learning process in the multi-agent case. Agents (i.e. firms) submit actions (i.e. probability of adjustment and reset markups) and the environment calculates the evolution of the state and exposes the transition tuple to the agents. The transition tuple is $\{r, s', s, a\}$, where r represents the reward of the agent in that period, s' represents new state for the agent, s represents the old state for the agent and a represents the action. The agents can use the tuple to improve their policy in a forward-looking manner.

We can use a simple duopoly game with stochastic marginal cost to illustrate the different elements. At period t , $n = 2$ firms indexed by i observe their marginal cost $c_{i,t}$ and decide the price $a_i \equiv p_i$ from an action space $A_{i,t} \equiv [c_{i,t}, c_{i,t} * 2]$. After both firms choose prices, they get rewards $r \equiv \pi(p_i, p_{-i})$ and the environment stochastically determines $S_{t+1} \equiv (c_{1,t+1}, c_{2,t+1})$ according to the transition function $p(r_t, S_{t+1} | S_t, (p_{1,t}, p_{2,t}))$. In this case, the transition function is calculated using the profit function (which is deterministic) and the stochastic process for the marginal cost.

The main challenge for multi-agent games is that from an agent's perspective, the environment is not stationary, even if the underlying game is. At the beginning of the game, all agents are experimenting and the dynamics of the system may be chaotic. After agents start to improve their actions, the strategies settle and the game becomes more predictable. Convergence guarantees can only be proved for a limited set of zero-sum games. In more general settings, there are no theoretical guarantees of convergence, and each game must be studied experimentally in order to observe the emergent strategies.

While multi-agent applications of RL are more recent than single-agent applications, there are many success cases that indicate that Deep RL algorithms can arrive at sophisticated strategies that require high degrees of long-term planning. A prominent example that uses the same algorithm and neural net architecture as we use is [Berner et al. \[2019\]](#), which achieves super-human performance at the game of Dota 2, one of the most popular and competitive multi-agent games in e-gaming. The game is challenging for multiple reasons. It is multi-agent, stochastic, it has a large action and observation space, and it requires long term strategy.

3 Model

Starting point: Mongey (2021) with general n firms. Time is discrete. There are two types of agents: households and firms. A unit measure of identical households consume goods, supply labor and own equal shares on all firms. There are (very large) N^s sectors indexed by j . In each sector, there are N^f firms indexed by i . Goods are differentiated first across, and then within sectors. Good ij is produced using labor with constant returns to scale. There are three types of shocks. First, there is an aggregate shock to the growth rate g_t of the money supply M_t . Second, there are idiosyncratic shocks to preferences for each good z_{ijt} . Third, in each period, firm ij draws a menu cost $\xi_{ijt} \sim H(\xi)$.

Households. Let $\mathbf{S}$ be the aggregate state and s_j be the sectoral state. The problem of consumers is:

$$\mathbf{V}(\mathbf{S}, B) = \max_{c_i(s), N, B'(S)} \log(C) - N + \beta \mathbb{E}[\mathbf{V}(\mathbf{S}', B(\mathbf{S}'))], \quad \text{where} \quad (2)$$

$$C = \left[\sum_{j=1}^{N^s} \left(\frac{1}{N^s} \right)^{\frac{1}{\theta}} \mathbf{c}(s_j)^{\frac{\theta-1}{\theta}} \right]^{\frac{\theta}{\theta-1}}, \quad \mathbf{c}(s_j) = \left[\sum_{i=1}^{N^f} \left(\frac{2}{N^f} \right)^{\frac{1}{\eta}} (z_i(s_j) c_i(s_j))^{\frac{\eta-1}{\eta}} \right]^{\frac{\eta}{\eta-1}} \quad (3)$$

subject to the nominal budget constraint

$$\sum_{j=1}^{N^s} \left[\sum_{i=1}^{N^f} p_i(s, \mathbf{S}) c_i(s) \right] + \int_S Q(\mathbf{S}, \mathbf{S}') B'(\mathbf{S}') d\mathbf{S}' \leq W(\mathbf{S})N + B(\mathbf{S}) + \Pi(\mathbf{S}) \quad (4)$$

The role of money comes from the fact that nominal expenditure needs to be paid with money:

$$\sum_{j=1}^{N^s} \left[\sum_{i=1}^{N^f} p_i(s, \mathbf{S}) c_i(s) \right] = PC = M \quad (5)$$

As Golosov and Lucas [2007] shows, under the preferences we have assumed this is equivalent to money in the utility functions, since the first order conditions for labor are linear in M . The demand for each good is:

$$d_{ij} = \frac{2}{N^f} z_{ij}^{\eta-1} \left(\frac{p_{ij}}{P_j} \right)^{-\eta} \left(\frac{P_j}{P} \right)^{-\eta} C \quad (6)$$

where

$$P_j = \left[\sum_{i=1}^{N^f} \frac{2}{N^f} \left(\frac{p_{ij}}{z_{ij}} \right)^{1-\eta} \right]^{\frac{1}{1-\eta}} \quad P = \left[\sum_{j=1}^{N^s} \frac{p_j^{1-\theta}}{N^s} \right]^{\frac{1}{1-\theta}} \quad (7)$$

The first order condition that characterizes the labor supply is

$$\frac{W}{P} = - \frac{U_n(C, N)}{U_C(C, N)} \rightarrow W = PC = M \quad (8)$$

The aggregate monetary shock g to the growth rate of money, such that $M' = gM$. The evolution of g follows

$$\log(g') = (1 - \rho_g) \log(\bar{g}) + \rho_g \log(g) + \sigma_g \epsilon'_g, \quad \epsilon_g \sim N(0, 1) \quad (9)$$

where ρ_g represents the persistence of the shock and $\log \bar{g}$ pins down the long-run growth in money supply.

Firms. Firms choose whether to move price ϕ_{ij} and, if they move, the reset price $p_{i,j}^*$, to maximize intertemporal profits. Quality shocks z_{ij} affect both preference and marginal cost. The evolution of z follows:

$$\log(z'_{ij}) = \log(z_{ij}) + \sigma_z \epsilon'_{ij}, \quad \epsilon_{ij} \sim N(0, 1) \quad (10)$$

Menu costs are paid in units of labor and each period, each firm draws a menu cost ξ from uniform and i.i.d. distribution $\xi_{ijt} \stackrel{i.i.d.}{\sim} U[0, \bar{\xi}/N^s]$.

For the case of $N^f = 2$, the recursive representation of the game is:

$$V_i(s, \mathbf{S}) = \int \max\{V^{adj}(s, \mathbf{S}) - W(\mathbf{S}) \frac{2\xi_i}{N^f N^s}, V_i^{stay}(s, \mathbf{S})\} dH(\xi_i), \quad (11)$$

$$V_i^{adj}(s, \mathbf{S}) = \max_{p_i^*} \gamma_{-i} \{\pi_i(p_i^*, p_{-i}^*, s, \mathbf{S}) + \mathbb{E}[Q(\mathbf{S}, \mathbf{S}') V_i(s'_{adj}, \mathbf{S}')]\} \quad (12)$$

$$+ (1 - \gamma_{-i}) \{\pi_i(p_i^*, p_{-i}, s, \mathbf{S}) + \mathbb{E}[Q(\mathbf{S}, \mathbf{S}') V_i(s'_{adj}, \mathbf{S}')]\} \quad (13)$$

Next, we are going to use several features of the problem to reduce the game to a markup game in which the firm observes the following states:

- Sector state s_j : "No-reset" markups of firms in your sector $s_{jt} = \{\bar{\mu}_{ijt}\}_i$

$$\bar{\mu}_{ijt} = \frac{\mu_{ijt-1}}{g_{t-1} e^{\sigma_z \epsilon_{ijt-1}}}$$

- Aggregate state $\mathbf{S}$: money growth g_{t-1} and aggregate markup μ_{t-1} , so $\mathbf{S}_t = \{\mu_{t-1}, g_{t-1}\}$.

The Markups formulation. Following Mongey [2021], we are going to express the problem in terms of markups. In particular, we define firm, sector and aggregate markup as:

$$\underbrace{\mu_{ij} = \frac{p_{ij}}{W/z_{ij}}}_{\text{Firm}}, \quad \underbrace{\mu_j = \frac{P_j}{W}}_{\text{Sector}}, \quad \underbrace{\mu = \frac{P}{W} = \frac{1}{C}}_{\text{Aggregate}}$$

Given these definitions, the aggregators in Equation 7 can be written as

$$\mu_j = \left[\sum_{i=1}^{N_j^f} 2 \frac{\mu_{ij}^{1-\eta}}{N^f} \right]^{\frac{1}{1-\eta}}, \quad \mu = \left[\sum_{j=1}^{N^s} \frac{\mu_j^{1-\theta}}{N^s} \right]^{\frac{1}{1-\theta}}$$

Also, we can express profits in terms of markups with the final objective of representing the entire recursive game in terms of markups. Given that we are working with a discrete number of sectors instead of a continuum of measure one, we need to normalize profits, so menu costs remain proportional when the economy grows, or we change the number of firms and sectors:

$$\pi_{ij} \frac{N^s N^f}{2W} = \tilde{\pi} = \left(\frac{\mu_{ij}}{\mu_j} \right)^{-\eta} \left(\frac{\mu_j}{\mu} \right)^{-\theta} \frac{1}{\mu} (\mu_{ij} - 1)$$

Using these aggregates and normalizations, we can frame the firm's problem as choosing reset markup μ_{ij}^* and probability of adjustment γ_{ij} . A firm that chooses a level γ_{ij} will move if the menu cost is lower $\xi_{ij} < \gamma_{ij} \bar{\xi}$, so the probabilities chosen coincides with the probability mass of the uniform menu cost distri-

bution.

Let $v_i(s, \mathbf{S}) = V_i(s, \mathbf{S})N^f N^s / 2W$. The equivalent recursive representation in the case of $N^f = 2$ is:

$$v_i(\mu_i, \mu_{-i}) = \int \max \left\{ v_i^{adj}(\mu_i, \mu_{-i}, \mathbf{S}) - \xi_i, v_i^{stay}(\mu_i, \mu_{-i}, \mathbf{S}) \right\} dH(\xi_i), \quad (14)$$

$$V_i^{adj}(\mu_i, \mu_{-i}, \mathbf{S}) = \max_{\mu_i^*} \left\{ \gamma_{-i} \left\{ \tilde{\pi}_i(\mu_i, \mu_{-i}, \mathbf{S}) + \mathbb{E} \left[v_i \left(\frac{\mu_i}{g' e^{\sigma_z \epsilon'_i}}, \frac{\mu_{-i}^*}{g' e^{\sigma_z \epsilon'_{-i}}}, \mathbf{S}' \right) \right] \right\} \right. \quad (15)$$

$$\left. + (1 - \gamma_{-i}) \left\{ \tilde{\pi}_i(p_i^*, p_{-i}^*, s, \mathbf{S}) + \mathbb{E} [Q(\mathbf{S}, \mathbf{S}') V_i(s'_{adj}, \mathbf{S}')] \right\} \right\} \quad (16)$$

The game. The reinforcement learning process can be represented as follows:

- *Environment Reset*:: The game starts with random markups: $\mu_{ij} \sim U[1, 2]$ and we get initial draws of ϵ_{ij} and ϵ_g .
- After constructing the initial state, the loop is:
 1. *Agent Act*: For $t \geq 0$: Given an observation of state $\{\bar{\mu}_{it}, \bar{\mu}_{-it}, \mu_{t-1}, g_{t-1}\}$, the agent chooses a reset markup rate μ_{it} and probability of adjusting γ_{it} according to stochastic policy $a_t(\mu_{it}, \mu_{-it}, \mu_{t-1}, g_{t-1})$.
 2. *Environment Step*: Given s_t , the environment calculates $\tilde{\pi}_{it}$, updates the states and exposes the following to the agents:
 - Reward: $r_t = \tilde{\pi}(\mu_i, \mu_j, \mu)$
 - Next State: $\bar{\mu}_{it+1} = \frac{\mu_{it}}{g_t e^{\sigma_z \epsilon_{it}}}$ and the aggregate (μ_t, g_t) .
 3. *Learn*: After observing one (or many) transitions

$$\text{transition} = \{\tilde{\pi}_{it}, \{\bar{\mu}_{it+1}, \bar{\mu}_{-it+1}, \mu_t, g_t\}, \{\bar{\mu}_{it}, \bar{\mu}_{-it}, \mu_{t-1}, g_{t-1}\}, \{\mu_{it}^*, \gamma_{it}^*\}\}$$

the agent updates the parameters of its policy $a(S)$ using RL.

Calibration. My starting point is [Mongey \[2021\]](#)'s calibration. The logic of the calibration is the following. The long run growth of money $\log \bar{g}$ is calibrated to match 2.5% inflation between 1985 and 2016. The persistence and size of monetary ρ_g and σ_g are calibrated to match the response of consumption to identified monetary shocks. The inter-sectoral elasticity of substitution η is set to match direct empirical evidence. The size of idiosyncratic shocks σ_z and menu costs ξ are calculated to match the frequency and size of price adjustment in retail price data. The intra-sectoral elasticity of substitution η is set to match average markups.

Table 1: Parameters

Parameter		Value
Within-sector elasticity of substitution	η	10.5
Across-sector elasticity of substitution	θ	1.5
Long-run increase money Supply	$\log \bar{g}$	0.021
Persistence monetary policy	ρ_g	0.61
Size of monetary shock	σ_g	0.0019
Size of idiosyncratic shocks	σ_z	0.0038
Upper-bound of Menu Cost	$\xi \in [0, \bar{\xi}]$	0.17
Monthly discount rate	β	$0.95^{1/12}$

Note: The calibration is based on [Mongey \[2021\]](#)

4 Results

The two main versions of the economy that we will study are a finite horizon version and an infinite horizon version. In the finite horizon version, the game has 24 normal periods (months), plus a final stage with 12 months of flexible prices with zero menu cost. This implementation mirrors the algorithm used in [Mongey \[2021\]](#), in which we start from the assumption that static Nash will be played forever starting next period and then solve by backward iteration. The infinite horizon game, on the other hand, is implemented as an episodic game with 60 months (5 years) but in which agents do not see calendar time, so they do not know when the game ends or whether the game ends at all, which introduces the possibility of cooperation. This episodic implementation of the infinite horizon game mirrors the implementation of our real business cycle benchmark, in which agents got the infinite horizon optimal solution by playing a large episodic game without knowing calendar time.

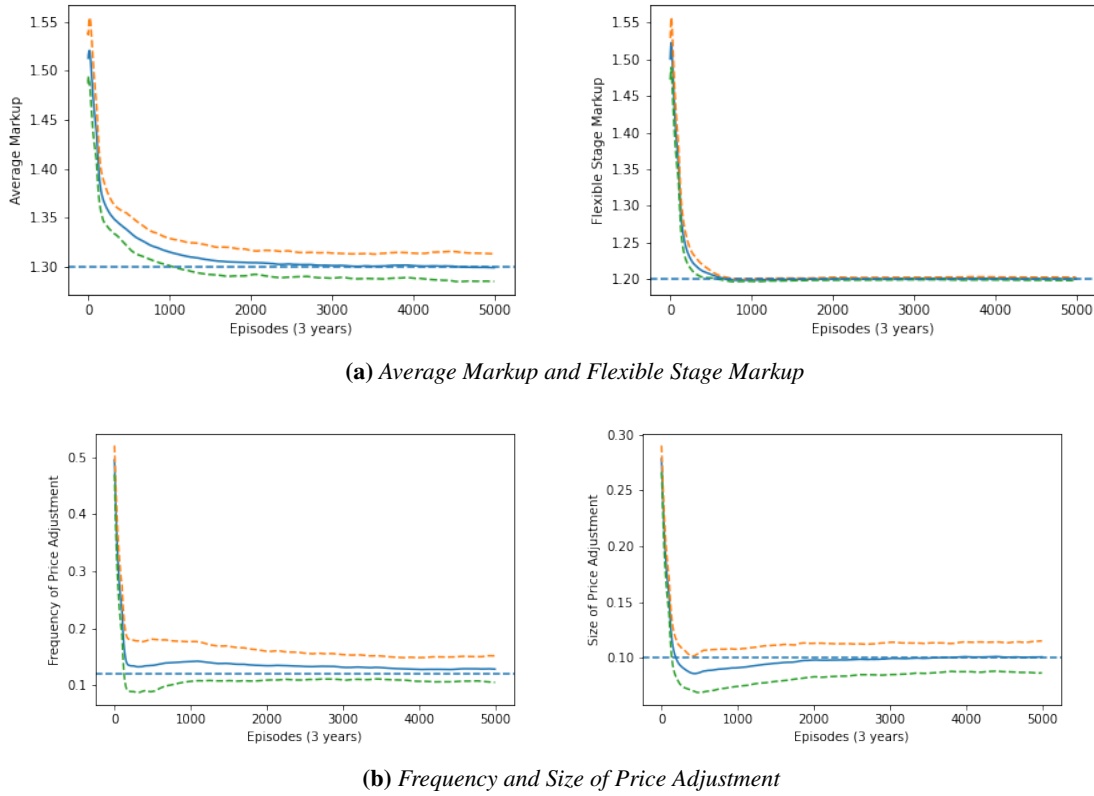
4.1 Experimental Settings and Convergence

The experimental setting is the following. In each economy, we start with random neural nets defined over a continuous support (so we are not discretizing the state space). Thus, in the beginning, the algorithms will act randomly and collect experiences. After 10 years (120 months), all the collected experiences from all firms are processed by the algorithm in order to update the neural nets. Since we have 200 firms, in each update of the neural nets the algorithms processes $200 \times 120 = 24000$ individual transitions. The optimization step is done using the Proximal Policy Optimization algorithm [[Schulman et al., 2017](#)], detailed in Appendix A. After 5000 updates of the neural net, we verify that the behavior has settled and recover the resulting policy parameters to further inspect and simulate the economy. During training, agents use a stochastic policy, which ensures that there is always some exploration. Upon learning, the algorithm will decrease the entropy of the distribution by an optimal amount to balance exploitation of previous knowledge with exploration. In

total, each neural net is updated with 60 million individual transitions. For each version of the economy, we repeat the learning experiment 50 times to check the range of converging strategies.

In Figure 5, we observe the aggregate statistics on markups and price adjustments during the learning phase. In the horizontal line, we have the statistics calculated with the exact solution in Mongey [2021] for the benchmark case of two firms, finite horizon, and Markovian strategies. As we can see, the statistics converge to the exact solution after approximately 5000 updates of the neural net. As a sanity check, we can calculate the average markup in the final stage with flexible prices and evaluate whether it converges to the flexible price Nash equilibrium which is 1.2. Notably, the markups in the flexible prices stage, shown in the right figure of panel (a), converge rapidly to the static Nash equilibrium of the game, which indicates that the algorithm identifies that after the last period with nominal rigidities the game changes to a static game and find the equilibrium predicted by theory.

Figure 5: Markups and Price Statistics Convergence



Note: Dashed horizontal line represents results under exact solution

4.2 Markups and Price Statistics

Table 2 shows the results of these three experiments, as a comparison to Mongey [2021]’s result. In column (2), we see that in the finite game, average statistics are similar to Mongey [2021], presented in column

(1). In particular, we observe slightly less frequent but larger adjustments, and a markup of 1.291 versus 1.3. Also, it is worth noticing that the standard deviation of markups in the finite version indicates a narrow range of converging markups in the finite game, in contrast with the infinite horizon game.

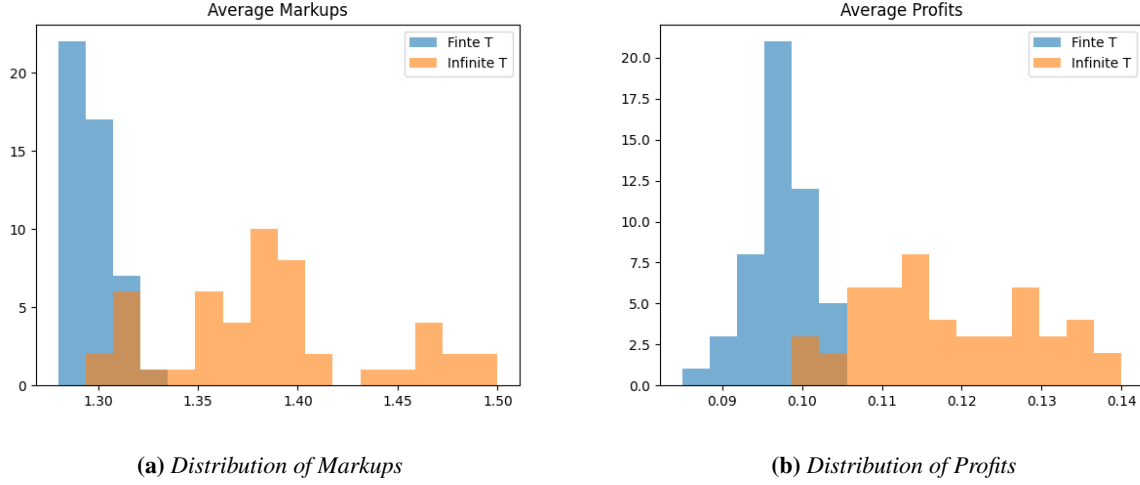
Table 2: *Results vs Mongey (2021)*

Moment	Mongey (2021)	Finite T	Infin. T	Min Inf. T	Max Inf. T
	(1)	(2)	(3)	(4)	(5)
Average Markup:	1.3	1.291 (0.009)	1.388 (0.055)	1.301	1.495
Frequency of Adj.:	0.12	0.115 (0.004)	0.131 (0.05)	0.11	0.149
Size of Adj:	0.106	0.11 (0.001)	0.098 (0.06)	0.071	0.105

Note: Column (1) shows the results in [Mongey \[2021\]](#). Column (2) shows the average result of 50 experiments for the finite horizon game, and the standard deviations are shown parenthesis. Column (3) shows the average result of 50 experiments for the infinite horizon game, and the standard deviations are shown parenthesis. Column (4) shows the result of the infinite horizon experiment with minimum-markup among the 50 experiments. Column (5) shows the result of the infinite horizon experiment with maximum-markup among the 50 experiments.

In the infinite horizon game, presented in column (3), we observe higher and more dispersed average markups, consistent with a higher range of equilibria supported by repeated game strategies. Also, on average, firms make more frequent adjustments. For the two versions of the economy, we will select the specific converging policy that has the minimal euclidean distance to the average statistics for the three moments presented in table 2, so we get a representative converging policy. From now on, we will refer to the finite horizon convergent strategy as the low-markup strategy, and the infinite horizon convergent strategy as the high-markup strategy.

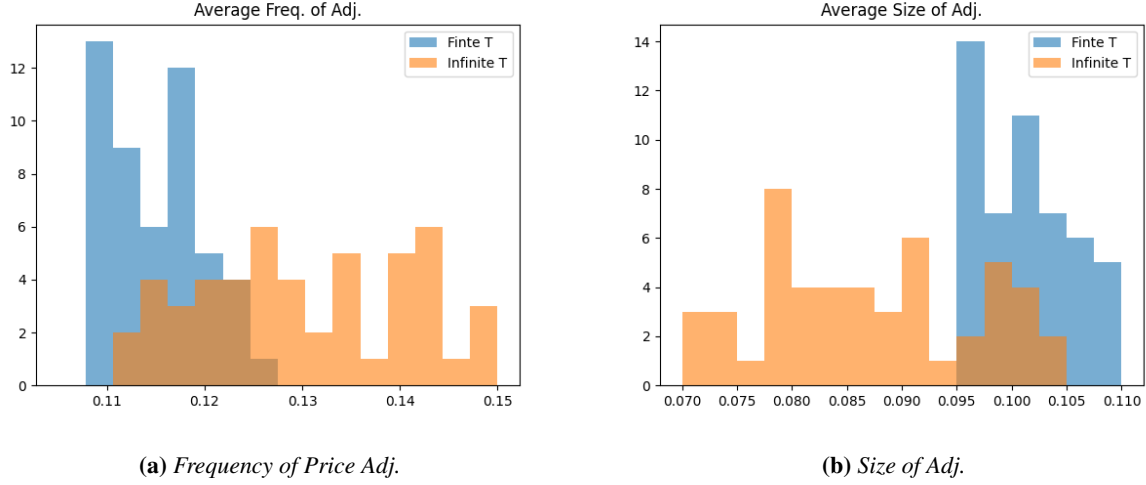
Figure 6: *Distribution of average Markups and Profits across observed equilibria*



Note: This figure shows the histogram of Average Markups and Average Profits for the 50 experiments in the finite horizon game (in blue) and the infinite horizon game (in orange). Markups are defined as price over marginal cost, defined in equation 3. Profits are defined as a percentage of marginal cost, defined in Equation 3.

To better characterize the diversity of converging equilibria, in Figures 6 and 7 we see histograms of the resulting markups, profits, frequency of adjustment, and size of adjustment. We see that the larger portion of the mass of results goes from Mongey [2021]'s statistics to equilibria with higher markups and slightly more frequent adjustments. Additionally, we note that the mode of the markup is 1.38, indicating that the number of converging equilibria with high markup is larger. As a reference, the full collusive markup for the game is 3, which is significantly higher than the maximum tacit collusive markup we observe. This is a consequence of the elasticities of substitution. Within sectors, the elasticity is high ($\eta = 10.5$), which indicates strong sectoral competition. In contrast, the inter-sectoral elasticity ($\theta = 1.5$), is much lower, so the full collusive equilibrium, in which a cartel competes only with the other sectors, would set a much higher markup. Overall, we can conclude that the range of collusive equilibria we observe corresponds to mild levels of collusion. This is consistent with the fact that this game exhibits many of the features that have shown to impede collusion in simpler games, such as idiosyncratic and aggregate shocks and nominal rigidities. In terms of profit rates, in panel (b) we see that the range of profit rates is between 9% and 13%.

Figure 7: Distribution of Frequency of Adjustment and Size of Adjustment



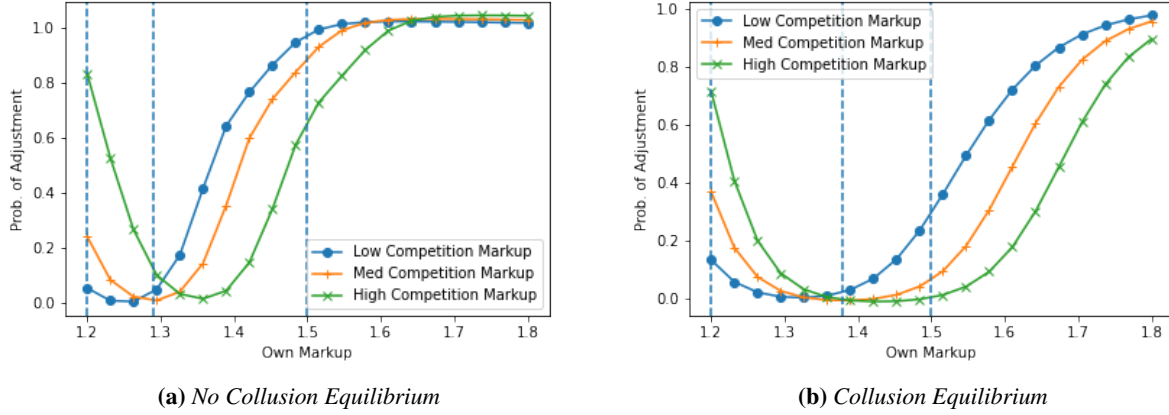
Note: This figure shows the histogram of Frequency of Price Adjustment and Size of Adjustment for the 50 experiments in the infinite horizon game. Frequency of Price Adjustment is defined as the proportion of agents that adjusted prices in a period: $\mathbb{E}[1\{p_{it} \neq p_{it-1}\}]$. Size of Adjustment is defined as the absolute value of log price changes: $\mathbb{E}[\log(p_{it}/p_{it-1})]$.

4.3 Inspecting the policies

With the goal of understanding the different behavior in collusive versus non-collusive strategies, we analyze the convergent policies in selected finite horizon equilibrium and infinite horizon equilibrium. For each equilibrium, there are two equilibrium strategies: probability of adjustment and reset markup. In the finite horizon game, the equilibrium strategies depend on four states: own markup, markup of the competition, money growth in the previous period, and aggregate markup in the previous period. In the infinite horizon game, the strategies will also condition on an internal state of the LSTM neural net that synthesizes the information from the past 12 months. When we inspect the policies, we will change one state at a time while leaving the other states at their average. Of all the policy-state combinations, we will focus on the ones that elucidate the response of agents to monetary shock. In particular, when there is positive shock to money growth the states change in two ways. First, we have a negative impact on the markup of all firms, which is reflected in the sectoral markups and aggregate markups. Second, agents are going to observe the higher money growth directly in the next period, which is going to give them a signal about future money growth.

To begin, we analyze how the optimal probability of adjustment varies with the no-reset markup of the firm, which we denote own markup to distinguish it from the no-reset markup of the competition. In Table 8, we see the results in both the non-collusive and the collusive equilibrium. To further inspect the policy we distinguish between three cases: when the competition has low markup ($\mu_{-i} = 1.2$), medium markup ($\mu_{-i} = 1.3$ for finite horizon and $\mu_{-i} = 1.38$ for infinite horizon) and high markup ($\mu_{-i} = 1.5$). Let's start our analysis from the no-collusion equilibrium and medium competition markup case (the

Figure 8: Probability of Adjustment as a Function of Current Markup



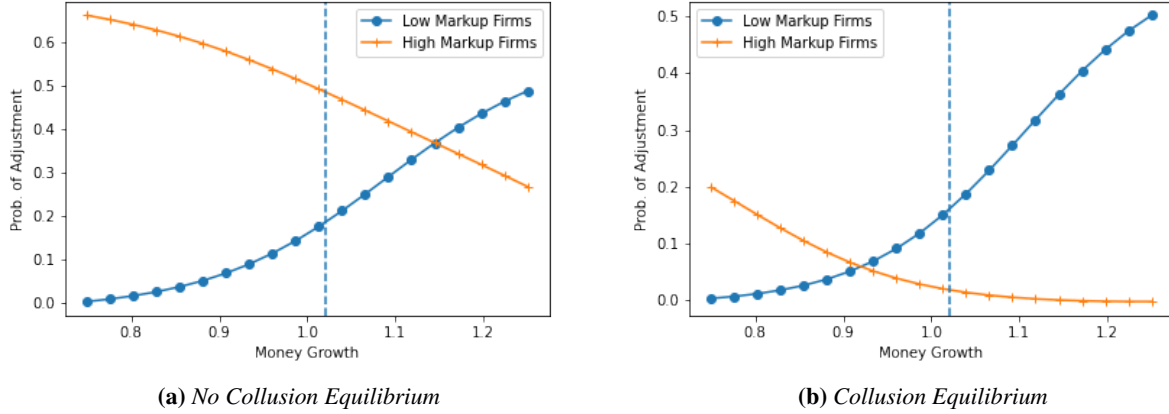
Note: This figure shows the probability of optimal adjustment γ_{ijt}^* as function of the "no reset" markup with which you enter the period $\bar{\mu}_{ijt}$. Dashed vertical lines represent the markup of competition for the three cases representing the three lines: the "Low Competition Markup" case ($\mu_{-i} = 1.2$) is represented by the blue line, the "Average Competition Markup" case ($\mu_{-i} = 1.3$ for finite horizon and $\mu_{-i} = 1.38$ for infinite horizon) is represented by the orange line, and the "High Competition Markup" case ($\mu_{-i} = 1.5$) is represented by the green line. Panel (a) shows the "No Collusion equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments. Panel (b) shows the "collusion equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments.

orange line) in panel (a) of Figure 8). We see that if the competition has a markup of 1.3, the probability of adjustment has an u-shape with a minimum in 1.3, which indicates that 1.3 for both firms is a low adjustment, or resting, point. When the competition has higher (lower) markup, the low adjustment point is higher (lower), indicating the presence of strategic complementarities, since the fact that your competition has higher markup incentives you to reset at a higher markup. The results in panel (b) show that, in the collusive equilibrium, the resting points are at a markup of 1.38. More broadly, the probability of adjustment policies serve as a sanity check that the agents have learned intuitive policies. Also, we observe strong nonlinearities in the policies that coincide with the expected behavior in this game, which are an indication that the methodology we are employing is able to capture non-trivial nonlinear policies.

We now turn to the analysis of the direct response to money growth. As discussed, the observation of a higher money growth in the last period is a signal of high money growth in the future, since monetary shocks are persistent (see Equation 9). In Figure 9 we see the probability of adjustment as a function of money growth. The behavior is consistent with the results in the literature. For low markup firms, a positive monetary shock moves their markup in the opposite direction as their desired markup, and higher expected growth means that further adjustment is necessary. Thus, for low-markup firms, the probability of adjustment depends positively on money growth. In contrast, for high markup firms, the monetary shock moves their markup in their desired direction, so their incentives to adjust diminishes. In panel (b), we see a similar response for the collusive equilibrium. At the average money growth, the impact of the slope of the probability of adjustment of the high-markup firms is lower than in panel(a). This has a simple explanation: high-markup firms ($\mu_i = 1.5$) are close to their desired markup, so shocks do not significantly alter their

incentives to adjust.

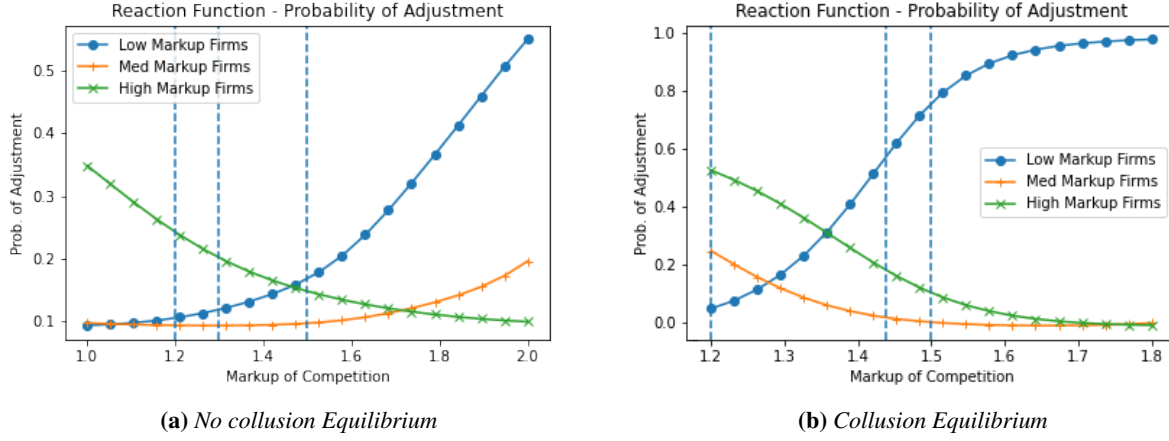
Figure 9: Probability of Adjustment as a Function of Money Growth



Note: This figure shows the optimal probability of adjustment γ_{ijt}^* as function of the money growth in the last period the period g_{t-1} . The dashed vertical line represents average money growth of $\bar{g} = 1.0221$. The "Low Markup" case ($\mu_i = 1.2$) is represented by the blue line, and the High Markup" case ($\mu_i = 1.5$) is represented by the orange line. Panel (a) shows the "No Collusion equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments. Panel (b) shows the "collusion equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments.

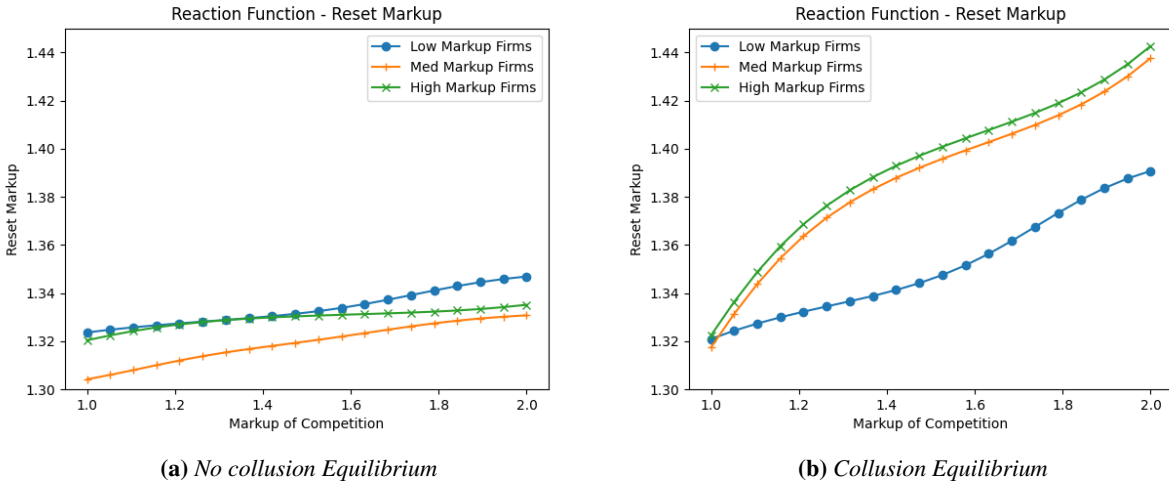
In Figure 10 we see the reaction function to changes in the markup of competition. This policy function allows us to understand the behavioral response of firms to pricing changes in the competition, which plays a fundamental role in our interpretation of the high-markup equilibrium as a tacit collusive equilibrium. In panel (a), we see that incentives to adjust behave as one would expect. In particular, we see that low markup firms increase their probability of adjustment when the other firm has a higher markup, since a higher competition markup increases the profit of adjusting your own markup upwards. For high-markup firms, we observe the opposite: the incentives to adjust decrease as the markup of the competition increases, as the profits gain of adjusting your own markup downwards decreases. In panel (a), the basic structure is the same, but we now observe that for firms with medium and high markups, incentives to adjust increase more sharply when the markup of the competition goes from the 1.4 range to the 1.2 range, which indicates that in the collusive equilibrium deviations from the high-markup equilibrium trigger a response in the following periods. It is important to notice that the response is triggered probabilistically depending on the realization of the menu-costs, which increases the temporary gain from deviation and may explain why we only observe mild levels of tacit collusion.

Figure 10: Reaction Function for Probability of Adjustment



Note: This figure shows the optimal probability of adjustment γ_{ijt}^* as function of the no-reset markup of the competition $\bar{\mu}_{-ijt}$. The dashed vertical line represents average money growth of $\bar{g} = 1.0221$. The "Low Competition Markup" case ($\mu_{-i} = 1.2$) is represented by the blue line, the "Average Competition Markup" case ($\mu_{-i} = 1.3$ for finite horizon and $\mu_{-i} = 1.38$ for infinite horizon) is represented by the orange line, and the "High Competition Markup" case ($\mu_{-i} = 1.5$) is represented by the green line. Panel (a) shows the "No Collusion equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments. Panel (b) shows the "collusion equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments.

Figure 11: Reaction Function for Reset Markup



Note: This figure shows the optimal reset markup μ_{ijt}^* as function of the no-reset markup of the competition $\bar{\mu}_{-ijt}$. Panel (a) shows the "No Collusion equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments. Panel (b) shows the "collusion equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments.

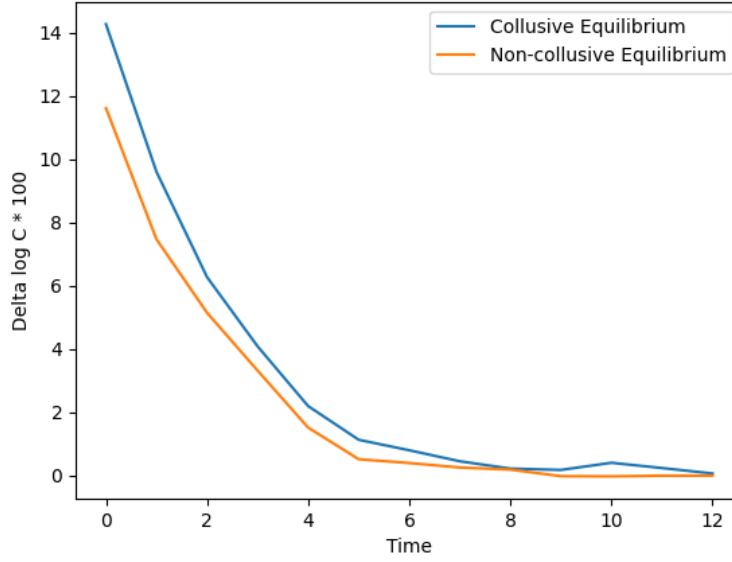
In figure 11 we see the reset-markup as a function of current markup, which can be interpreted as the best-response function to a lagged change in the markup of the competition. As we can see, this reaction

function has a positive but low slope in the non-collusive equilibrium. For medium-markup firm, if we focus on the case in which the markup of the competition is at 1.3, which we know is a resting point from figure 9, we observe an almost flat reaction function. In contrast, for the collusive equilibrium, we see that all firms have a steep reaction function. Complemented with the reaction function for the probability of adjustment in figure 10, we get a clear picture: in the collusive equilibrium firms react more to changes in the markup of the competition. This implies that prices are complementary and that such complementarity is dynamic in the sense that the reaction will unfold over time. Strategically, this steeper reaction function increases the cost of deviating from a high-markup equilibrium, since firms know that if they move to lower markup the other firm will respond with higher probability of adjustment (figure 10) and a lower reset markup (figure 11).

4.4 Collusion and Effectiveness of Monetary Policy

In order to evaluate the effectiveness of monetary policy, we simulate the economy under both equilibrium policies for a total of 1000 years. Importantly, for each equilibrium, we simulate the economy with and without the monetary shocks, but with the same idiosyncratic shocks. The simulation without the monetary shocks allows us to control for the aggregate variation induced by idiosyncratic shocks, which is necessary since we have a discrete number of firms. Specifically, we subtract the log consumption of the stationary economy to the economy with monetary shocks. Then, we calculate the standard deviation of the filtered log consumption for the entire simulation, which is the preferred measure of monetary policy effectiveness in the literature. With this estimate, we can compare the effectiveness of monetary policy in the non-collusive equilibrium versus the collusive equilibrium. We get that monetary policy effectiveness in the collusive equilibrium is 0.057 versus 0.041 in the non-collusive, which implies that collusion increases effectiveness by up 42% increase. Also, we regress the changes in the filtered log consumption on lags of monetary shocks. In figure 12, we observe the obtained impulse responses. At impact, the effect of monetary policy on log consumption is 20% higher on the collusive equilibrium. The effect is also slightly more persistent, which explains the cumulative effect on the economy's volatility.

Figure 12: *Impulse response of filtered log consumption to lagged monetary shocks*



Note: this figure shows the coefficients β_τ of a regression of log changes in consumption in lags of the monetary shock: $\Delta \log C_t = \alpha + \beta_\tau \epsilon_{t-\tau}^g + \eta_t$. We run one of these regressions for each lag τ . The data comes from simulating 12000 months of an economy in which firms act according to their converging policies. The blue line shows the "No Collusion Equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments. The orange line shows the "Collusion Equilibrium", which is defined as the finite horizon experiment with statistics that are closer to the average statistics, out of the 50 experiments.

4.5 Revisiting the Effect of Concentration

In Table 3, we see how changing the number of firms from in each sector alters the average markup, the price adjustment statistics and the effectiveness of monetary policy. For each number of firms $N^f \in \{2, 3, 4, 5\}$ we first present the results for the finite horizon game, with no collusion, and we compare them with the result under collusive strategies. In addition to the aforementioned increased effectiveness of monetary policy under collusive strategies, the results show two clear patterns. First, in the non-collusive case we observe that more firms result in lower markups, more price adjustment and a slightly less effective monetary policy. Second, the degree of collusion, measured as the difference between average markups in the collusive versus non-collusive case, goes down as the number of firms goes up. As a consequence of these two patterns, we observe that higher concentration has strong effects on the collusive case since collusion is easier among fewer firms, which increases effectiveness. For example, going from 3 to 2 firms increase effectiveness only by 5% in the non-collusive case, versus 25% for the collusive case. The mild effect of concentration in the no collusion case is consistent with the theoretical findings of Wang and Werning [2020], which show in a stylized framework with Calvo pricing that under the preferences we have assumed here the number of firms have a weak effect on the half-life of the monetary shock. Our results under more

general repeated game strategies suggest that the effect of concentration can be strong if fewer firms result in higher levels of observed tacit collusion.

Table 3: *Number of firms and Price Dynamics*

Moment	2 firms		3 firms		4 firms		5 firms	
	No Coll	Coll.	No Coll.	Coll.	No Coll.	Coll.	No Coll.	Coll.
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Average Markup:	1.291	1.388	1.212	1.234	1.185	1.202	1.173	1.188
Frequency of Adj.:	0.115	0.131	0.125	0.132	0.132	0.136	0.138	0.141
Size of Adj:	0.098	0.106	0.095	0.108	0.109	0.110	0.112	0.117
Effectiveness of MP	0.50%	0.71 %	0.49%	0.56%	0.38 %	0.41%	0.38%	0.39%

Note: Column (1) to (8) show the average result of 50 experiments for each combination of number of firms and with versus without collusion. The no collusion columns show the average results for the finite horizon columns, while the collusion columns show the results for the infinite horizon game.

5 Conclusions

We have characterized the effectiveness of monetary policy in a general equilibrium model with menu costs and oligopolistic competition at the sector level. After showing that our methodological approach renders the same results as the recent literature for a specific tractable case, we generalize the number of firms per sector and the set of strategies that firms can play. Our main findings are that collusion increases the effectiveness of monetary policy and that market concentration, by facilitating collusion, also increases the real effect of monetary policy. By inspecting the converging strategies, we reveal that the increased effectiveness of monetary policy in the collusive case is due to stronger reactions of firms to the price of the competition. Such strong reactions induce large punishments in case firms deviate from the high-markup equilibrium.

Methodologically, the paper presents a new approach to studying general equilibrium models with strategic agents. The results emphasize the relevance of starting from a known framework, and then extending as needed. Without a benchmark, the convergent strategies of artificial intelligence algorithms are hard to interpret meaningfully. The particular application that we have discussed, in which we have a discrete but large number of firms, highlights two classes of models that would benefit from this approach. The first, more thoroughly discussed in this paper, is dynamic general equilibrium models with strategic agents. The second class of economies that can be studied with Deep Reinforcement Learning is models with granular, or large, firms. In our baseline framework we have 400 firms in 200 sectors. Since our interest is in the aggregate response to monetary policy, we have not introduced specificity to the problem of any particular sector. Nevertheless, the fact that even at such a scale, we are still from the computational limit of the methodology suggests that models with a diverse type of large firms are feasible.

References

- Fernando Alvarez and Francesco Lippi. Price setting with menu cost for multiproduct firms. *Econometrica*, 82(1):89–135, 2014.
- John Asker, Chaim Fershtman, and Ariel Pakes. Artificial intelligence and pricing: The impact of algorithm design. Technical report, National Bureau of Economic Research, 2021.
- Marlon Azinovic, Luca Gaegauf, and Simon Scheidegger. Deep equilibrium nets. *Available at SSRN 3393482*, 2019.
- Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.
- Mark Bils and Peter J Klenow. Some evidence on the importance of sticky prices. *Journal of political economy*, 112(5):947–985, 2004.
- Ariel Burstein and Christian Hellwig. Prices and market shares in a menu cost model, 2007.
- Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolo, and Sergio Pastorello. Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10):3267–97, 2020.
- Dan Cao, Wenlan Luo, and Guangyu Nie. Global dsge models. *Available at SSRN 3569013*, 2020.
- Mingli Chen, Andreas Joseph, Michael Kumhof, Xinlei Pan, Rui Shi, and Xuan Zhou. Deep reinforcement learning in a monetary model. *arXiv preprint arXiv:2104.09368*, 2021.
- Matias Covarrubias, Germán Gutiérrez, and Thomas Philippon. From good to bad concentration? us industries over the past 30 years. *NBER Macroeconomics Annual*, 34(1):1–46, 2020.
- Chaim Fershtman and Ariel Pakes. Dynamic games with asymmetric information: A framework for empirical work. *The Quarterly Journal of Economics*, 127(4):1611–1661, 2012.
- Mikhail Golosov and Robert E Lucas. Menu costs and phillips curves. *Journal of Political Economy*, 115(2):171–199, 2007.
- Philipp Grohs, Arnulf Jentzen, and Diyora Salimova. Deep neural network approximations for monte carlo algorithms. *arXiv preprint arXiv:1908.10828*, 2019.
- Jonathon Hazell, Juan Herreno, Emi Nakamura, and Jón Steinsson. The slope of the phillips curve: evidence from us states. Technical report, National Bureau of Economic Research, 2020.
- Byoung Jun and Xavier Vives. Strategic incentives in dynamic duopoly. *Journal of Economic Theory*, 116(2):249–281, 2004.

- Mahdi Ebrahimi Kahou, Jesús Fernández-Villaverde, Jesse Perla, and Arnav Sood. Exploiting symmetry in high-dimensional dynamic programming. Technical report, National Bureau of Economic Research, 2021.
- Nobuhiro Kiyotaki and Randall Wright. On money as a medium of exchange. *Journal of political Economy*, 97(4):927–954, 1989.
- Paul R Krugman. Pricing to market when the exchange rate changes, 1986.
- Per Krusell and Anthony A Smith, Jr. Income and wealth heterogeneity in the macroeconomy. *Journal of political Economy*, 106(5):867–896, 1998.
- Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- Lilia Maliar, Serguei Maliar, and Pablo Winant. Deep learning for solving dynamic economic models. *Journal of Monetary Economics*, 2021.
- Ramon Marimon, Ellen McGrattan, and Thomas J Sargent. Money as a medium of exchange in an economy with artificially intelligent agents. *Journal of Economic dynamics and control*, 14(2):329–373, 1990.
- Eric Maskin and Jean Tirole. A theory of dynamic oligopoly, ii: Price competition, kinked demand curves, and edgeworth cycles. *Econometrica: Journal of the Econometric Society*, pages 571–599, 1988.
- Virgiliu Midrigan. Menu costs, multiproduct firms, and aggregate fluctuations. *Econometrica*, 79(4):1139–1180, 2011.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- Simon Mongey. Market structure and monetary non-neutrality. Technical report, National Bureau of Economic Research, 2021.
- Emi Nakamura and Jon Steinsson. Monetary non-neutrality in a multisector menu cost model. *The Quarterly journal of economics*, 125(3):961–1013, 2010.
- Julio J Rotemberg and Michael Woodford. Oligopolistic pricing and the effects of aggregate demand on economic activity. *Journal of political Economy*, 100(6):1153–1207, 1992.
- John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- John N Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.
- Olivier Wang and Iván Werning. Dynamic oligopoly and price stickiness. Technical report, National Bureau of Economic Research, 2020.
- Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. King’s College, Cambridge United Kingdom, 1989.

A Deep Reinforcement Learning Algorithms

In this appendix, we evaluate the potential of reinforcement learning in solving a simple RBC model and compare the results with the solutions obtained using GDSGE, a toolbox that solves DSGE models with global methods.

A.1 A Simple RBC Model

Time is discrete. The representative agents solve

$$\max \mathbb{E} \sum_{t=0}^{\infty} \beta^t u(c_t) = \mathbb{E} \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma}$$

The representative firms have production technology defined by

$$Y_t = f(z_t, K_t, L_t) = z_t K_t^\alpha L_t^{1-\alpha} = z_t K_t^\alpha$$

where L_t is constant and normalized to 1, and z_t denotes the productivity shock imposed following a two-point Markov process regulated by the transitional matrix

$$\begin{pmatrix} \pi_{LL} & 1 - \pi_{LL} \\ 1 - \pi_{HH} & \pi_{HH} \end{pmatrix}.$$

And investment technology performs according to

$$K_{t+1} = (1 - \delta) K_t + I_t$$

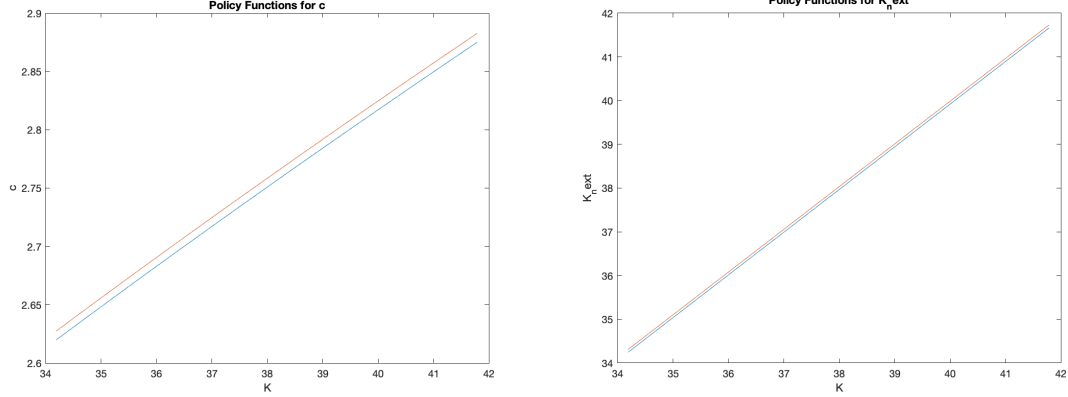
Finally, the market clearing conditions are given by $c_t + I_t = Y_t$. This and equations (1.13) and (1.11) give the state-transition function:

$$K_{t+1} = (1 - \delta) K_t + z_t K_t^\alpha - c_t$$

This environment yields the necessary and sufficient optimality condition summarized in the Euler Equation:

$$c_t^{-\sigma} - \beta \mathbb{E}_t \{ [\alpha z_{t+1} K_{t+1}^{\alpha-1} + (1 - \delta)] c_{t+1}^{-\sigma} \} = 0$$

Solving RBC with Policy Iteration. The following results are generated using the GDSGE toolbox [Cao et al., 2020]. The figures show the solved policy function for c , i.e. $c = h(K)$ with a range of values for K corresponding to the two shock levels.



GDSGE is a toolbox that provides global solutions to the nonlinear Dynamic Stochastic General Equilibrium models based on the Simultaneous Transition and Policy Function Iteration (STPFI) algorithm developed in [Cao et al. \[2020\]](#). This algorithm jointly solves for policy and transition functions over the iterations to obtain recursive equilibria. It mobilizes dynamic libraries that implement the actual computations efficiently in C++, while the models can be intuitively and straightforwardly coded in scripts on a Matlab-based interface.

A.2 Policy Gradient Algorithms

Proximal Policy Optimization (PPO) [[Schulman et al., 2017](#)] method is a state-of-art version of policy gradient methods supported by the company *OpenAI*⁹. It aims to strike a balance between ease of implementation, sample complexity and ease of tuning. PPO tries to obtain an update at each step that minimizes the cost function while ensuring that the deviation from the previous policy is relatively small to avoid performance collapse.

There are two primary variants of PPO methods:

- Adaptive-KL Penalty includes a term that penalizes the KL-divergence in the objective function when evaluating an update.
- Clipped Surrogate Objective method relies on specialized clipping in the objective function to avoid a large variation in the updated policy.

The second variant is most widely applied in RL given that clipping the change in the policy distribution is much cheaper computationally than attempts to solve an optimization problem with KL-divergence penalties or constraints.

⁹See <https://spinningup.openai.com/en/latest/> for an introduction to state-of-the-art algorithms made by OpenAI.

Vanilla Policy Gradient. Reinforcement learning concerns finding an optimal behavior strategy, or policy, that gives the agent the optimal rewards in a given environment. Instead of learning the state/action value function and then selecting actions, the **policy gradient** methods directly target and optimize the policy. The policy, $\pi_\theta(a|s)$, is usually a parametrized function with respect to θ . There are various versions of the algorithms that can be applied to estimate parameters θ to maximize discounted rewards.

Denote $a \in \mathcal{A}$ actions, $s \in \mathcal{S}$ states, and

$\pi_\theta(a|s)$ the stochastic policy parametrized by θ , with $\pi_\theta(a|s) = \mathbb{P}_\pi[A = a|S = s]$;

$d^\pi(s)$ the stationary distribution of Markov chain for π_θ , with $d^\pi(s) = \lim_{t \rightarrow \infty} \mathbb{P}[s_t = s|s_0, \pi_\theta]$;

$V^\pi(s)$ the value of state s under the policy π ;

$Q^\pi(s, a)$ action-value function that accesses the expected return under (s, a) when following a policy π ;

$A(s, a)$ advantage function, $A(s, a) = Q(s, a) - V(s)$.

The reward function is

$$J(\theta) = \sum_{s \in \mathcal{S}} d^\pi(s) V^\pi(s) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q^\pi(s, a)$$

where all π refers to the same parametrized policy π_θ . Policy-based methods are naturally more useful in the continuous space, where there is an infinite number of actions or states. It is computationally too expensive to estimate the values for the continuous action and state spaces. To bypass this requirement, policy gradient methods move θ toward the direction suggested by the gradient $\nabla_\theta J(\theta)$ to acquire the best θ for π_θ . Computing gradient $\nabla_\theta J(\theta)$ is difficult because it depends on both the action selection and the stationary distribution of the state following the target selection. In principle, we need to estimate the effect on the state distribution by a policy update. To bypass this expensive step, we rely on the *policy gradient theorem*, which reforms $\nabla_\theta J(\theta)$ to eliminate the derivative of $d^\pi(\cdot)$ and simplify the computation.

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} Q^\pi(s, a) \pi_\theta(a|s) \quad (17)$$

$$\propto \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} Q^\pi(s, a) \nabla_\theta \pi_\theta(a|s) \quad (18)$$

Next, we present the proof of the policy gradient theorem¹⁰. The gradient of the state value function can be unrolled as

¹⁰The proof presented here follows Sutton and Barto [2018].

$$\begin{aligned}
\nabla_{\theta} V^{\pi}(s) &= \nabla_{\theta} \left(\sum_{a \in \mathcal{A}} \pi_{\theta}(a | s) Q^{\pi}(s, a) \right) \\
&= \sum_{a \in \mathcal{A}} (\nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi}(s, a) + \pi_{\theta}(a | s) \nabla_{\theta} Q^{\pi}(s, a)) \\
&= \sum_{a \in \mathcal{A}} \left(\nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi}(s, a) + \pi_{\theta}(a | s) \nabla_{\theta} \sum_{s', r} P(s', r | s, a) (r + V^{\pi}(s')) \right) \\
&= \sum_{a \in \mathcal{A}} \left(\nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi}(s, a) + \pi_{\theta}(a | s) \sum_{s', r} P(s', r | s, a) \nabla_{\theta} V^{\pi}(s') \right) \\
&= \sum_{a \in \mathcal{A}} \left(\nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi}(s, a) + \pi_{\theta}(a | s) \sum_{s'} P(s' | s, a) \nabla_{\theta} V^{\pi}(s') \right)
\end{aligned}$$

Denote $\phi(s) := \sum_{a \in \mathcal{A}} \nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi}(s, a)$ and the transition probability from state s to state s' after k steps under policy π as $\rho^{\pi}(s \rightarrow s', k)$. Then it follows simply that

$$\begin{aligned}
\rho^{\pi}(s \rightarrow s', k=1) &= \sum_a \pi_{\theta}(a | s) P(s' | s, a) \\
\rho^{\pi}(s \rightarrow x, k+1) &= \sum_{s'} \rho^{\pi}(s \rightarrow s', k) \rho^{\pi}(s' \rightarrow x, 1)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\nabla_{\theta} V^{\pi}(s) &= \phi(s) + \sum_a \pi_{\theta}(a | s) \sum_{s'} P(s' | s, a) \nabla_{\theta} V^{\pi}(s') \\
&= \phi(s) + \sum_{s'} \sum_a \pi_{\theta}(a | s) P(s' | s, a) \nabla_{\theta} V^{\pi}(s') \\
&= \phi(s) + \sum_{s'} \rho^{\pi}(s \rightarrow s', 1) \nabla_{\theta} V^{\pi}(s') \\
&= \phi(s) + \sum_{s'} \rho^{\pi}(s \rightarrow s', 1) \left[\phi(s') + \sum_{s''} \rho^{\pi}(s' \rightarrow s'', 1) \nabla_{\theta} V^{\pi}(s'') \right] \\
&= \phi(s) + \sum_{s'} \rho^{\pi}(s \rightarrow s', 1) \phi(s') + \sum_{s''} \rho^{\pi}(s \rightarrow s'', 2) \nabla_{\theta} V^{\pi}(s'') \\
&= \phi(s) + \sum_{s'} \rho^{\pi}(s \rightarrow s', 1) \phi(s') + \sum_{s''} \rho^{\pi}(s \rightarrow s'', 2) \phi(s'') + \sum_{s'''} \rho^{\pi}(s \rightarrow s''', 3) \nabla_{\theta} V^{\pi}(s''') \\
&= \sum_{x \in S} \sum_{k=0}^{\infty} \rho^{\pi}(s \rightarrow x, k) \phi(x)
\end{aligned}$$

where the last equality follows from repeatedly unrolling of $\nabla_{\theta} V^{\pi}(s''')$. Denote $\eta(s) := \sum_{k=0}^{\infty} \rho^{\pi}(s_0 \rightarrow s, k)$.

Then,

$$\begin{aligned}
\nabla_{\theta} J(\theta) &= \nabla_{\theta} V^{\pi}(s_0) \\
&= \sum_s \sum_{k=0}^{\infty} \rho^{\pi}(s_0 \rightarrow s, k) \phi(s) \\
&= \sum_s \eta(s) \phi(s) \\
&= \left(\sum_s \eta(s) \right) \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \phi(s) \\
&\propto \sum_s d^{\pi}(s) \sum_a \nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi}(s, a)
\end{aligned}$$

A.3 PPO algorithm.

The Proximal Policy Algorithm (PPO) enforces a clipped surrogate objective function to ensure that the policy update does not come in overly large steps. It updates policy via solving:

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s, a \sim \pi_{\theta_k}} [J^{CLIP}(s, a, \theta_k, \theta)]$$

Denote the ratio between old and new policies

$$r(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}$$

To impose a limitation on the distance, measured by $r(\theta)$, PPO adds the constraint by forcing $r(\theta)$ to stay bounded within a small interval around 1, i.e. $r(\theta) \in [1 - \epsilon, 1 + \epsilon]$, where ϵ is a pre-specified hyperparameter. Note that the **clip function** can clip the ratio to be within the chosen boundaries, $1 - \epsilon$ and $1 + \epsilon$. Thus, the objective can be written as

$$J^{CLIP}(\theta) = \mathbb{E} \left[\min \left(r(\theta) \hat{A}_{\theta_k}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\theta_k}(s, a) \right) \right]$$

where $\hat{A}(\cdot)$ is an estimate of the advantage function $A(\cdot)$.

When applying PPO on the network architecture with shared parameters for both policy and value functions, in addition to the clipped reward, the objective function is augmented with an error term on the value estimation and an entropy term to encourage sufficient exploration. That is,

$$J^{CLIP'}(\theta) = \mathbb{E} \left[J^{CLIP}(\theta) - c_1 (V_{\theta}(s) - V_{\text{target}})^2 + c_2 H(s, \pi_{\theta}(\cdot)) \right]$$

where c_1, c_2 are two hyperparameter constants.

Figure 13: Pseudo-code of the PPO algorithm

Algorithm 1 PPO-Clip

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go $\hat{R}_t$.
- 5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

- 7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

- 8: **end for**
-

Note: This pseudo-code was made available by the company that designed the algorithm:
<https://spinningup.openai.com/en/latest/algorithms/>

Pseudo-code. Figure 13 shows the pseudo-code of the algorithm. The computation of the advantage of taking an action on a particular step is done through Generalized Advantage estimation (GAE), which is explained in Subsection A.3.1.

Important PPO Hyperparameters:

- **Batch size:** the batch size is a hyperparameter of gradient descent that controls the number of training samples to work through before the model's internal parameters are updated.
- **Clip parameter:** clip is associated with the PPO-clip method which restricts the ration between old and new policies to prevent overstepping updates.
- **Time horizon:** the agent takes n steps, collects experience by interacting with the environment and uses that experience to update the policy. A short horizon indicates larger bias but lower variance, and a long horizon leads to higher variance and smaller bias.
- **Learning rate and learning rate schedule:** learning rate is set to specify the step size of the Adam optimizer for the gradient descent algorithm. The general finding is that with learning rate schedule

turned on, which provides a scheme of decay of learning rate in each trial, the path becomes more stable, with fewer fluctuations along the tail and better performance of convergence.

A.3.1 Generalized Advantage Estimation (GAE)

During the implementation of a PPO algorithm, the advantage function $A(\cdot)$ is usually unknown, so we compute a variance-reduced advantage function estimator using the Generalized Advantage Estimation (GAE) method [Schulman et al., 2015].

First define the temporal difference residual (TD residual): $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$. Given the current estimate of value function V , we can derive a class of advantage function estimators taking the form:

$$\begin{aligned}\hat{A}_t^{(1)} &= r_t + \gamma V(s_{t+1}) - V(s_t) \\ \hat{A}_t^{(2)} &= r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t) \\ &\dots = \dots \\ \hat{A}_t^{(\infty)} &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots - V(s_t)\end{aligned}$$

where r_t is the period reward, and γ is a parameter specific to the method. The choice of k in $\hat{A}_t^{(k)}$ reflects the trade-off between the variance and bias of the estimator, since a large k reflects a large variance but low bias, whereas a small k has a low bias and large variance. Using all the estimators above to strike a balance between the two, the Generalized Advantage Estimator (GAE) is introduced and defined as

$$\begin{aligned}\hat{A}_t^{\text{GAE}(\gamma, \lambda)} &:= (1 - \lambda) \left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots \right) \\ &= (1 - \lambda) \left(\delta_t^V + \lambda (\delta_t^V + \gamma \delta_{t+1}^V) + \lambda^2 (\delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V) + \dots \right) \\ &= (1 - \lambda) \left(\delta_t^V (1 + \lambda + \lambda^2 + \dots) + \gamma \delta_{t+1}^V (\lambda + \lambda^2 + \lambda^3 + \dots) \right. \\ &\quad \left. + \gamma^2 \delta_{t+2}^V (\lambda^2 + \lambda^3 + \lambda^4 + \dots) + \dots \right) \\ &= (1 - \lambda) \left(\delta_t^V \left(\frac{1}{1 - \lambda} \right) + \gamma \delta_{t+1}^V \left(\frac{\lambda}{1 - \lambda} \right) + \gamma^2 \delta_{t+2}^V \left(\frac{\lambda^2}{1 - \lambda} \right) + \dots \right) \\ &= \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}^V\end{aligned}$$

where both parameters γ and λ adjust the bias-variance trade-off associated with the resulting estimator. Note that the parameter $\lambda \in [0, 1]$ is more important and typically used for the adjustment.

A.4 Results and Comparison Between RL and GDSGE

We run several experiments in parallel to tune and implement the different parameter setups outlined above and study the respective outcomes. The default algorithm does relatively well, but we can get to 99.99% of

the optimal discounted rewards by doing intuitive modifications to the main parameters. By reducing the size of the steps when the algorithm is reaching its peak performance, we can fine-tune the optimization around the optimal point.

Case 1. Default algorithm. When using the default algorithm, we achieve a discounted reward of 97.5383 compared to 97.5989, the value calculated from the GDSGE solution.

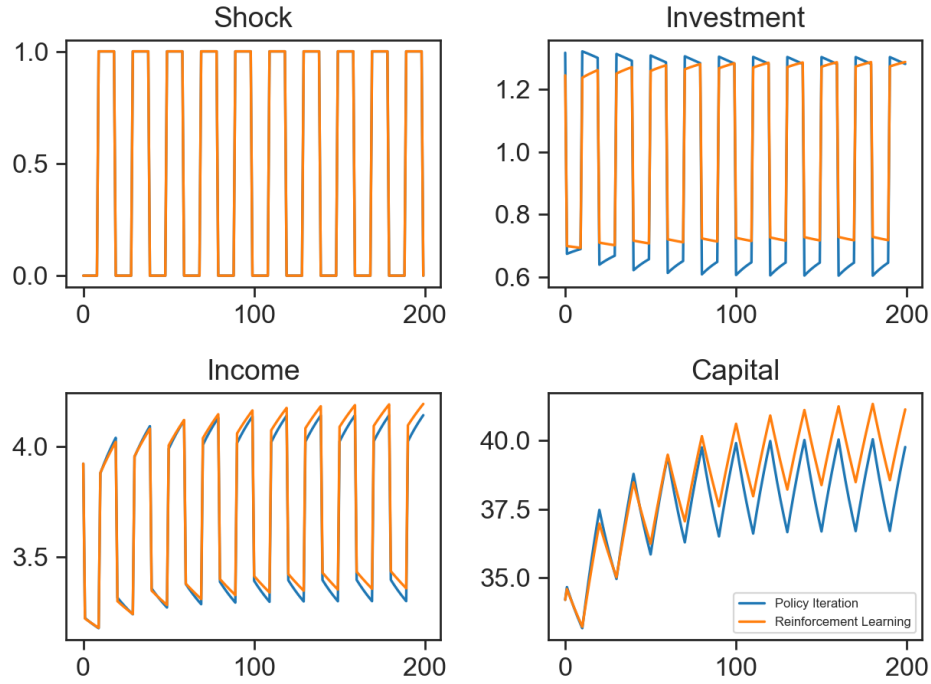


Figure 14: *Case 1: With learning rate decay.*

Note: This figure shows the performance of two algorithms in solving a Real Business Cycle Model with two productivity states. The orange line is the reinforcement learning solution and the blue line is the policy iteration solution. Given that we used a fine grid, the policy iteration solution can be interpreted as the exact solution. The upper left panel shows the shock. The upper right panel shows investment for both algorithms. The bottom left panel shows the income process and the bottom right panel shows capital level. The Policy Iteration solution was calculated using the GDSGE policy iteration framework.

To measure how well our training outcome compares to the real optimal solution achieved by the GDSGE method, we simulated an alternative path based on random selection of policies. It achieved a discounted reward of 86.32 over 400 periods. So the improvement brought by the trained policy as a fraction of the improvement given by the real optimal policy is $(97.5383 - 86.32)/(97.5979 - 86.32) = 99.471\%$.

Case 2. With learning rate decay. When we allow the default rate to decay, so it takes smaller steps when it is close to the optimum, we achieved a discounted reward of 97.553 compared to 97.5979, the value calculated from the GDSGE solution, which represents a 99.526% efficacy.

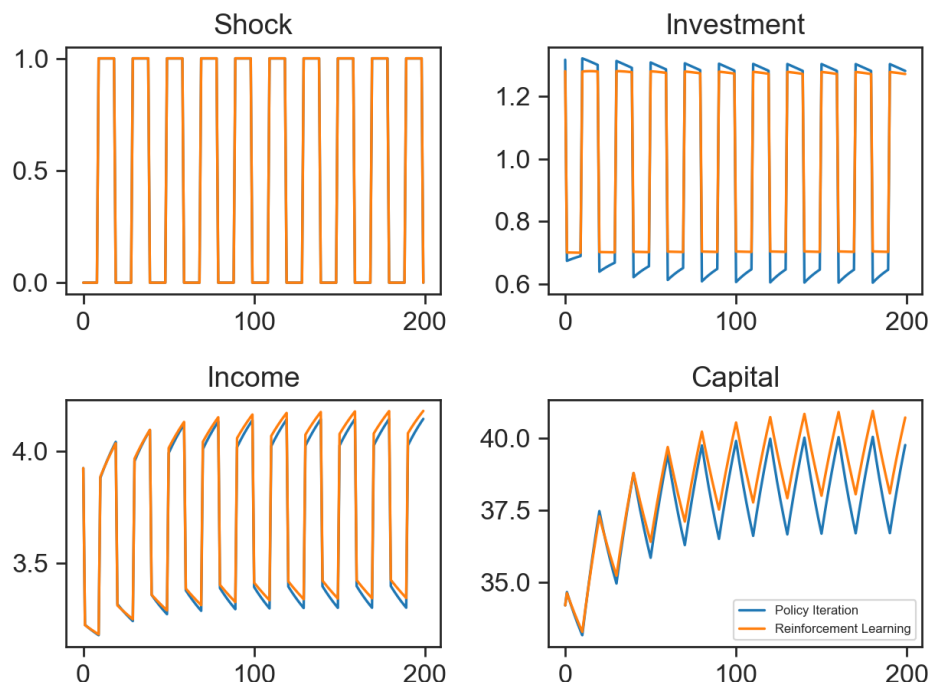


Figure 15: Case 2: With Learning rate Decay

Note: This figure shows the performance of two algorithms in solving a Real Business Cycle Model with two productivity state. The orange line is the reinforcement learning solution and the blue line is the policy iteration solution. Given that we used a fine grid, the policy iteration solution can be interpreted as the exact solution. The upper left panel shows the shock. The upper right panel shows investment for both algorithms. The bottom left panel shows the income process and the bottom right panel shows capital level. The Policy Iteration solution was calculated using the GDSGE policy iteration framework.

Case 3. With smaller step sizes and smaller neural net with layers size (128,128). When we clip the change in the distribution to 0.15 instead of 2 (see explanation of PPO's hyperparameters), and we reduce the size of the neural nets' layers to half, we achieve a discounted reward of 97.5939, compared to 97.5979, the value calculated from the GDSGE solution, which represents a 99.996% of accuracy.

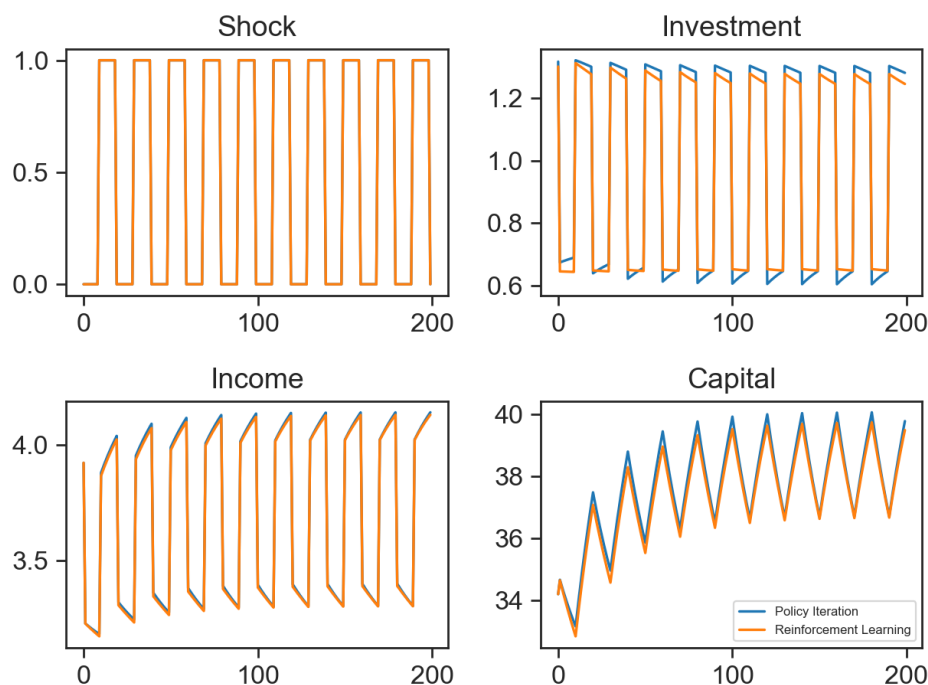


Figure 16: Case 3 and definitive: lower step size and smaller neural net

Note: This figure shows the performance of two algorithms in solving a Real Business Cycle Model with two productivity states. The orange line is the reinforcement learning solution and the blue line is the policy iteration solution. Given that we used a fine grid, the policy iteration solution can be interpreted as the exact solution. The upper left panel shows the shock. The upper right panel shows investment for both algorithms. The bottom left panel shows the income process and the bottom right panel shows capital level. The Policy Iteration solution was calculated using the GDSGE policy iteration framework.