Reinforcement learning induced non-neutrality of monetary policy in computational economic simulation

Bořivoj Vlk[1,*]

Abstract

In a Real Business Cycle model, monetary shock does not affect real variables, and economic agents are assumed to understand the model's structure. This article shows how it is possible to build a macroeconomic agent-based simulation from standard textbook Real Business Cycle model and how to utilize reinforcement learning to drive agents' decision making. The reinforcement learning algorithm of choice in this article is Q-learning, extended with fuzzy approximation. Q-learning is a simple algorithm based on incremental updates of estimated future rewards. As such, it circumvents introducing black boxes into the simulation and does not require strong assumptions on economic agents' rationality and expectations. This simulation falls into Real Business Cycle model category, but the reinforcement learning driven decision making mechanism of economic agents causes monetary policy to be non-neutral in the short run.

Keywords: Q-learning, agent-based modeling, agent-based computational economics, reinforcement learning, Real Business Cycle
JEL: C6, C7, E3, E4

1. Introduction

Drawing on insights from Dynamic Stochastic General Equilibrium Modeling (DSGE) and Agent-based Computational Economics (ACE) literature, this article presents an equilibrium model of fluctuations of aggregate economic variables.

The model is designed as a computational simulation. The simulation's source of inertia, and rigidity, is the reinforcement learning (RL) based decision making mechanism of economic agents. The simulation does not include sticky prices or wages. The idea is to take as much from Real Business Cycle (RBC) methodology as possible while omitting assumptions about agents' rationality.

Instead, agents' behavior drives the RL computer algorithm built around Bellman's optimality principle. The design of the simulation omits the strong-form rational expectations assumption and replaces it with constructive rationality as defined in Sinitskaya and Tesfatsion (2015).

For monetary shock to affect real variables in any model, some rigidity must be present. Without rigidities, an instantaneous price adjustment would offset the shock. Micro-foundations

*Corresponding author
Email address: borivoj.vlk@gmail.com (Bořivoj Vlk)

of DSGE models feature sticky nominal prices and wages. Usually, firms cannot reset their prices freely, but any adjustments are subject to Calvo-style frictions.(Calvo, 1983) Today, these models are fine-tuned for a wide range of applications and economies their authors aspire to model. Nonetheless, all of the DSGE models share similar building blocks. From the basic RBC framework, they take representative firm and household solving a dynamic optimization problem, i.e., time-dependent profit and utility maximization. On top of that, they also include various forms of rigidities and imperfections.

A classic model by Kollman (1996) features sticky nominal prices and wages, but there are many phenomena in natural economic systems for which DSGE modelers wish to account. Apart from sticky prices and wages, a widely known model by Smets and Wouters (2003) also contains external habit formation in consumption and capital utilization. A model by Mankiw and Reis (2007) creates inertia in aggregate variables without nominal rigidities but uses so-called *sticky information* - only a fraction of agents update their observations of price and wage levels each period. De Grauwe (2008) proposes a model in which agents use heuristic rules to forecast the inflation rate.

Different forms of financial frictions became widespread after the criticism of DSGE that followed the Great Recession. (Gertler and Kiyotaki, 2015) include incentives for agents to participate in shadow banking schemes. A pre-Great Recession model featuring a financial accelerator was developed by Bernanke et al. (1998), and many more examples exist. [1]

In some sense, this article also considers adding imperfections and rigidities to RBC. It is also not the first take on modeling economic systems around agents with RL decision making mechanisms.

Even when large processing power was not widely available, such algorithms were utilized on economic topics. One example is the take of Marimon et al. (1990) on the popular model of the emergence of currency by Kiyotaki and Wright (1989) using an algorithm somewhat similar to the one used throughout this article.

More recently, Sinitskaya and Tesfatsion (2015) compare performance (in terms of utility/profit obtained by firms and households) of several RL based decision making rules in a multi-agent macroeconomic model. They conclude that one of the critical factors is memory length. In other words, how much weight does the agent's decision making mechanism put on more recent or past observations of price and wage levels. This article utilizes a modified version of one of the algorithms examined by Sinitskaya and Tesfatsion (2015), namely Q-learning. Although they show that this algorithm is not the best performing, it will enable us to implement a market clearing mechanism that does not require explicit assumptions on how agents form their price and wage level expectations. We will also focus more on the dynamic properties of the model instead of comparing Q-learning to a social planner benchmark model.

The computational simulation based on RBC will exclude some components of standard methodology (aside from the already mentioned strong-form rational expectation).

Firstly, the economy will be modeled as N-player game. When modeling perfectly competitive markets, a continuum of players is used to capture the negligence of the influence of any individual agent.(Aumann, 1964) The N-player game approach utilized in this article is

[1] Detailed review can be found, for example, in Christiano et al. (2018).

necessary given the ACE's fundamental nature. The N-player game does not capture the idea of a perfectly competitive market as well as a continuum of players. However, with the wide availability of computational power, a modeler can choose N to be quite large, and therefore the individual's impact on the market is minimized.

Secondly, the simulation presented in this article is populated by heterogeneous agents. The representative agent (RA) assumption will be replaced by the assumption that each agent is characterized by parameter vector $\boldsymbol{\theta}_t$. The heterogeneity is endogenous as the value of vector $\boldsymbol{\theta}_t$ is identical for all agents at the start. However, as the simulation progresses, it evolves for each agent separately.

We will also drop the rational expectations assumption as agents' expectations are implicitly captured in parameter vector $\boldsymbol{\theta}_t$.

The sum of individual agents' preferences will produce aggregate demand and supply functions. Nevertheless, as opposed to the methodology which uses representative agents in DSGE, aggregated individuals' behavior(or behavior, if you will) captured in $\boldsymbol{\theta}_t$ may form multiple equilibria. (see for example Sonnenschein (1972)) Appendix A presents details on an algorithm that derives aggregate supply and demand functions and sets the equilibrium price of goods. A variety of ACE paperss utilizing many different decision making mechanisms studied the effects of heterogeneity and expectations on economic systems. Dosi et al. (2020) present a model based on Keynes + Schumpeter extended for heterogeneous expectations formed by several rules, from which agents select based on their predictive performance. Expectations of a fraction of agents in the New Keynesian model by Jump et al. (2019) are formed based on past observed values, and the remainder of agents are fully rational. See Tesfatsion (2017), or Fagiolo and Roventini (2017) for a review of how some of the standard assumptions, like RA and rational expectations, may be unrealistic and how a decision making mechanism based on reinforcement learning (and ACE in general) could address this issue.

In the last decade, reinforcement learning algorithms, especially multi-layer neural networks, saw wide use across different fields. Such algorithms have gained tremendous popularity with the emergence of open-source parallel processing frameworks. This article utilizes an algorithm called Q-learning, developed by Watkins and Dayan (1992). It is a relatively simple algorithm, and one might ask, why not use some neural network or any of the deep reinforcement learning algorithms.

However, macroeconomic policymaking needs to be transparent, and any model that hopes to become a practical tool can not feature black boxes. Algorithms, such as Q-learning, built around Bellman's principle of optimality, are simple. To such an extent that agents' behavioral patterns can be visualized in a graph or even evaluated by simple calculations done by hand.

Adding a reinforcement learning mechanism into the RBC framework is another way of including rigidities. The resulting model should have some similar properties with models that include sticky prices and wages. We will show how reinforcement learning can be a source of rigidity and, therefore, be a source of non-neutrality of monetary policy and coordination failures.

The rest of this article is organized as follows: Section 2 presents the basic model setting. Section 3 introduces the reader to the reinforcement learning algorithm utilized in agents' decision making mechanism, and section 4 presents computational simulation results and

shows how the model responds to a monetary shock.

2. Model setting

A typical dynamic RBC framework is a model of an economy that assumes representative households and representative firms solving a dynamic (time-dependent) optimization problem. Generally, the model assumes discrete time. Both households and firms are price/wage takers. The model also assumes that efficient market hypothesis (EMH) holds and, crucially - perfect foresight strong-form rational expectations.
The simulation presented in this article is built around a simple textbook model, for example, as presented in Galí (2008). As opposed to the original model setting, our simulation includes not only the modified household's budget constraint, but the firm's budget constraint is also introduced. The necessity of this will be explained in later sections when we introduce market clearing, i.e., price setting mechanism. On top of that, the firm's production function will include both labor and capital, for which it will pay costs in the form of wages and costs of opportunity, respectively.

2.1. Household's optimization problem

In the typical RBC framework, households maximize their expected lifetime utility.

$$\max_{\{C_t, L_t\}_{t=1}^{\infty}} E \sum_{t=1}^{\infty} \beta^t U(C_t, L_t) \tag{1}$$

Where C_t is quantity consumed of single good and L_t denotes labor supplied and $\beta \in (0, 1)$ is a discount factor, which represents time preference. The utility function $U(C_t, L_t)$ has the satisfy following properties:

$$\frac{\partial U(C_t, L_t)}{\partial C_t} > 0 \quad \text{and} \quad \frac{\partial^2 U(C_t, L_t)}{\partial C_t^2} \leq 0 \tag{2}$$

$$\frac{\partial U(C_t, L_t)}{\partial L_t} \leq 0 \quad \text{and} \quad \frac{\partial^2 U(C_t, L_t)}{\partial L_t^2} \leq 0 \tag{3}$$

and in our simulation will be defined as

$$U(C_t, L_t) = \frac{C_t^{1-\sigma}}{1-\sigma} - \frac{L_t^{1+\varphi}}{1+\varphi} - Cl \tag{4}$$

where $Cl > 0$ is cost of living. In each period, household faces budget constraint. Therefore, the optimization problem needs to be solved for each $t > 1$ subject to:

$$P_t C_t + F_t M_t = M_{t-1} + W_{t-1} L_{t-1} + D_{t-1} \tag{5}$$

where P_t is price of consumption good. W_t denotes nominal wage, D_t are dividend payments received and M_t represents the quantity of one-period, nominally riskless bonds purchased in period $t-1$ and maturing in period t. Each bond pays one unit of money at maturity, and its price is F_t.

The right side of the equation 5 will be referred to as the household's budget and denoted B_t for the rest of this article.
It is important to note that the budget constraint defined here differs from the original model in Galí (2008). First, we assume that the households spend all their funds on either consumption or purchasing bonds. This assumption allows us to have just two control variables: consumption level C_t and labor supplied L_t. Therefore, it decreases the dimensionality of the optimization problem. The budget constraint is, therefore, equality.
Second, for a unit of labor supplied in period $t-1$, households will not receive wage immediately, but in period t together with dividend payments and firms' profits determine the dividend payments.
Lastly, since our model is a numerical simulation with a finite number of rounds and some initial state, households will be endowed with finite money holdings Ie_h and for $t = 1$ following budget constraint must hold:

$$P_1 C_1 + F_1 M_1 = Ie_h \tag{6}$$

2.2. Firm's optimization problem

Firm seeks to maximize its profits each period t by using labor and capital to produce goods it sells at a given price P_t. The firm's production function is given by:

$$Y_t = A L_t^{1-\alpha} K_t^{\alpha} \tag{7}$$

Where $\alpha \in (0, 1)$, L_t denotes labor hired, K_t is capital input and A is a technology parameter which we will keep constant. In this article, we focus on monetary shocks, and keeping A constant allows us to reduce the number of state variables.
As a firm maximizes its profits in each period t, it solves the following optimization problem:

$$\max_{\{L_t\}_{t=1}^{\infty}} E \sum_{t=1}^{\infty} \beta^t (P_t Y_t - W_t L_t - K_t(\frac{1}{F_t} - 1)) \tag{8}$$

Where $K_{t-1}(1/F_{t-1} - 1)$ is opportunity cost of capital. Price P_t and wage W_t are given and therefore are state variables. The firm is subject to the following budget constrain for $t > 1$:

$$L_t W_t + K_t = P_{t-1} Y_{t-1} + K_{t-1} - W_{t-1} L_{t-1} - K_{t-1}(\frac{1}{F_{t-1}} - 1) - D_{t-1} \tag{9}$$

Where D_{t-1} is dividend. Again, as with the household's budget constraint, the right side of the equation will be referred to as the firm's budget and denoted B_t. A firm's budget constraint is given by equality, and therefore we can have just one control variable, which is L_t - the amount of labor hired. Firms are at $t = 0$ endowed with finite initial money holdings Ie_f, and budget constraint is given by

$$N_1 W_1 + K_1 = Ie_f \tag{10}$$

2.3. State and control

Assuming that both firms and households are price and wage takers, variables P_t, W_t, F_t and B_t are state variables. From both the firms and households perspective, P_t, W_t and F_t can follow some stochastic process, while B_t is influenced by their past actions. L_t is a control variable for both firms and households.

Let us assume that there is a finite number of possible values of L_t. This assumption can be justified by the fact that there is a finite number of households in the real economy. Each of them can supply a finite amount of labor and is willing to work only for some exact number of hours, therefore $L_t < \infty$ and $L_t \in \mathbb{N}_0$ for both households and firms. Since the household spends its current budget entirely on either consumption or purchase of bonds, we can write that household selects some value of control variable $c_t \in [0, 1]$, and then the amount of goods consumed is

$$C_t = \frac{c_t(M_{t-1} + W_{t-1}L_{t-1} + D_{t-1})}{P_t} \tag{11}$$

and number of bonds purchased is then

$$M_t = \frac{(1 - c_t)(M_{t-1} + W_{t-1}L_{t-1} + D_{t-1})}{F_t} \tag{12}$$

For household, a combination of values of c_t and L_t will be for the rest of this article referred to as *action* and will be denoted a. Firms action contains just the value of variable L_t. A set of all possible a forms a set of actions denoted $\mathcal{A}$. Also, combination of values of P_t, W_t, B_t and F_t will be for the rest of this thesis referred to as *state* and will be denoted s. A set of all possible states will be denoted $\mathcal{S}$.

To make things simpler, we will assume that $P_t \in (0, 1], F_t \in (0, 1]$ and $W_t = 1; \forall t \in \mathbb{N}$. This means that one unit of labor always buys at least one unit of consumption goods, and bonds have a non-negative yield. We can keep the wage constant without the loss of generality since only the W_t/P_t relation, i.e., real wage, is what concerns our model.

Now let's redefine the optimization problem in equations 1 and 8 in more general terms. Each period t, both households and firms observe some state $s \in \mathcal{S}$ and select action $a \in \mathcal{A}$. Then, from agents perspective, the state of the world moves to s' in $t + 1$ with probability $P_a(s, s')$ and agent receives reward, i.e. either utility or profit, denoted $R_a(s, s')$. Let π be a set of ordered state action pairs, $\pi = \{a_{s_0}, a_{s_1}, a_{s_2}, ...\}$. In this case, ordered means that a particular action is to be executed in a particular state. Such set π will be referred to as policy for the rest of this article.

This definition can be utilized in reformulating the household's/firm's ultimate goal - they seek cumulative reward maximizing policy. In other words, the agent seeks a set of decisions that maximizes the expected reward(utility/profit) over time $t = 1, ..., \infty$.

Using all of the above, we can rewrite the optimization problem for both firms and households as:

$$\max_{\pi \in \mathcal{M}} \sum_{t=1}^{\infty} \sum_{s' \in \mathcal{S}} \beta^t R_{a_t}(s, s') P_{a_t}(s, s') \tag{13}$$

Where $\mathcal{M}$ denotes the set of all possible *policies*. The solution of this problem will be denoted π^* and referred to as *optimal policy*.

2.3.1. Simulation schema

The simulation has to run for a finite number of periods n, with a finite number of firms and households that meet every period in the markets for goods and labor. The information about the actual value of n will not be provided to the agents. Algorithm 2.3.1 provides a general overview of how the simulation is structured and orchestrated in pseudo-code.

More detail about how agents take actions(including the random action selection) and up-

Algorithm 1 Simulation schema

Initialize N_f firms with arbitrary π_i and initial $B_{f,i} = I_{ef}$ for each firm $i = 1, ..., N_f$
Initialize N_h households with arbitrary π_i and initial $B_{h,i} = I_{eh}$ for each household $i = N_f + 1, ..., N_f + N_h$
Set P_t to arbitrary value from interval $(0, 1)$
Set F
$s_i \leftarrow < P_t, F, B_i >; \forall i = 1, ..., N_f + N_h$
for $t \leftarrow 1$ to n **do**
    Identify which agents will take random action this period
    Set P_t such that the difference between supply and demand on the goods market is minimized
    for $i \leftarrow 1$ to $N_f + N_h$ **do**
        $s'_i \leftarrow < P_t, F, B_i >$
        Receive $R_{a_i}(s_i, s'_i)$ and update π_i
        $s'_i \leftarrow s_i$
        $agent_i$ decides on action a_i
    end for
    Randomly assign households labor to firms
    Firms produce goods
    Households pay for goods to firms and consume purchased goods
    Households' utility and firms' profit get stored in $R_{a_i}(s_i)$, (s'_i is yet to be determined at the beginning of next iteration)
    Firms pay for labor and capital
    Firms pay to households for their labor and dividends get distributed
end for

date their policies is provided in section 3 and how the market clearing price is endogenously set is provided in Appendix A.

Simulation is designed such that F is an exogenous variable. This can be viewed as the central bank setting interest rates. In section 4, we will examine the effects of changing the value of F during the simulation on aggregate economic variables, such as consumption level and unemployment rate.

For the households, the reward $R_a(s, s')$ is equal to the value of the utility function. For the firms, the reward is the profit minus some cost of living Cl. Rewards are non-negative, i.e., if a firm makes a loss or a household's utility would be negative, its reward is set to 0.

It is possible for firms that use sub-optimal policy to be left with a minuscule amount of money holdings at the end of the period and be unable to hire more labor. In such a case, the firm is effectively removed from the simulation. Every period, such a fundless firm has a

chance of being *resurrected*, which means its policy π gets reset to the initial state. The firm receives the initial money endowment again.
Firms' dividends are equally distributed among all households. Dividends are paid as a portion of the firm's profit, which would make the firm's money holdings rise above a specific threshold. For example, if the threshold is 5, the firm's current money holdings are 4, the dividend rate is 50%, and the firm is about to receive a profit of 2, it would pay a dividend of 0.5.
The outlined structure of the simulation has been designed to be as simple as possible, but it is easy to extend. One can, for example, add an endogenous F setting mechanism - a central bank or a mechanism that would add more agents. For example, new firms might enter the market if the profits are high.

3. Decision making mechanism

The usual analytical solution of the model presented in the prior section is fairly simple[2]. However, the model solution's inter-temporal condition(14) requires us to assume agents have perfect foresight.

$$M_t = \beta E_t\{\frac{U(C_{t+1}, L_{t+1})/dC_{t+1}}{U(C_t, L_t)/dC_t}\frac{P_t}{P_{t+1}}\} \tag{14}$$

That is, their expectations about the value of P_{t+1} at period t are correct. In this article, we wish to drop the perfect foresight assumption, and therefore we need some other method of solution.

Real-world households and firms may not be able to solve the whole problem analytically and find π^*. When agents cannot judge complicated, risky alternatives against one other, they may result to using a simple trial-and-error method instead.[3] Let's assume, that the agent does not have the complete knowledge of probabilities $P_{a_t}(s, s')$, nor do they have complete knowledge of $R_{a_t}(s, s')$. Therefore, they can not directly learn the *optimal policy* using analytical methods. The lack of knowledge about probabilities $P_a(s, s')$ and rewards $R_a(s, s')$ prevents an agent from any attempt to evaluate rewards from possible actions before actually taking them.
Let us say the only information agent obtains is the value of the reward received and the resulting state. The sequence of rewards from executing different actions in different states and the sequence of transitions from one state to another forms the agent's experience.
Such experience then provides them with a set of heuristic rules - for each state, choose the action that previously brought the highest reward. However, such a rule may prove ineffective. Besides the current reward, the agent also needs to consider what future rewards they expect in the resulting state.
Sometimes it may be more beneficial to take an action that brings little or no reward now but leads to a state in which the agent can obtain a very high reward later. For example, working provides no immediate reward. However, working puts the agents in a state with more funds tomorrow, and in a state with more funds, a higher reward can be obtained by spending those funds. This concept of state transitions and delayed rewards can be grasped

[2]For an example of a complete analytical solution, see Galí (2008).
[3]More about human decision making under uncertainty can be found in Tversky and Kahneman (1974)

using Dynamic programming. (Bellman, 2010)

One RL computer algorithm built on top of Dynamic programming is called Q-learning. In this section, we will introduce *vanilla* Q-learning, which assumes that sets $\mathcal{A}$ and $\mathcal{S}$ are finite and enumerable, as well as so-called Fuzzy Q-iteration (Busoniu et al., 2010) which will allow for $\mathcal{S}$ to be continuous.
The description of the Q-learning algorithm that follows is based on Mitchell (1997).

3.1. Q-learning

The largest possible expected reward - i.e., the the expected reward, given that agent uses *optimal policy*, written as a function of the state, is called the alue function:

$$V(s) = \max_{\pi \in \mathcal{M}} \sum_{t=1}^{\infty} \sum_{s' \in \mathcal{S}} \beta^t R_{a_t}(s, s') P_{a_t}(s, s') \tag{15}$$

One of the core ideas behind dynamic programming is that finding the Value function is equivalent to finding the optimal policy - the solution to the optimization problem.
The value function can be found by utilizing Bellman's optimality principle, which, when stated mathematically, is in essence, an implicit definition of the value function.

$$V(s) = \max_a \{R_a(s, s') + \beta E[V(s')\} \tag{16}$$

The literature refers to this formal definition of Bellman's optimality as Bellman's equation. Q-learning is an iterative method of obtaining its solution.
The first step in deriving the Q-learning algorithm is redefining $V(s)$ in a more general form. Let's define a function $Q(s, a)$ as expected reward from executing action a in a state s and following optimal policy afterward.

$$Q(s, a) = E[R_a(s, s') + \beta V(s')] \tag{17}$$

This definition is called Q-function, and it is a mapping from state/action space to cumulative reward. Suppose an agent learns values of Q-function for all state and action pairs. In that case, he/she will be able to find an optimal policy without explicit knowledge of underlying rewards $R_a(s, s')$ and probabilities $P_a(s, s')$. Finding the optimal policy is equivalent to finding reward values for all state and action pairs because the Q-function is the exact quantity maximized in the definition of $V(s)$. Therefore, for the optimal action a^* in the current state following holds:

$$a^* = \arg\max_a Q(s, a) \tag{18}$$

Notice also this relation between the Q-function and the value function:

$$V(s) = \max_a Q(s, a) \tag{19}$$

Therefore, we can introduce a recursive definition of the Q-function. Notice the similarity to Bellman's equation.

$$Q(s, a) = E[R_a(s, s') + \beta \sum_{s' \in \mathcal{S}} P_a(s, s') Q(s', a')] \tag{20}$$

Agents learn the Q-function by gradually adjusting its estimate, commonly denoted $\hat{Q}$. At $t = 0$ agent initializes a table of estimates $\hat{Q}_0(a, s)$ for each state/action pair to arbitrary finite values. As the simulation progresses, in each period the agent chooses an action associated with the highest value of the estimate $\hat{Q}_t$ given the current state. To allow for some exploration, each period agent has a probability $\epsilon \in (0, 1)$ of choosing a random action. Occasionally selecting random action allows the agent to escape a trap of a local maxima of the Q-function.
After the agent executes this action, the state of the world changes from s to s', and the agent receives reward $R_{a_t}(s, s')$ and updates the estimate using the following rule:

$$\hat{Q}(s, a)_{t+1} \leftarrow (1 - l)\hat{Q}_t(s, a) + l(R_{a_t}(s, s') + \beta \max_a \hat{Q}_t(s', a)) \tag{21}$$

Parameter $l \in (0, 1)$ is learning rate. Given some additional technical assumptions, $\hat{Q} \to Q$ as $t \to \infty$ and therefore agent learns the optimal policy.(Watkins and Dayan, 1992) However, from a single agent's perspective, the rewards and transition probabilities constantly evolve in our model setting. The convergence issue cannot be viewed merely as in single agent learning a task. Given the market clearing mechanism our model uses, see Appendix A, P_{t+1} is determined from all values of estimates $\hat{Q}_t(s, a)$ of all agents. Therefore, as with the real economy, the full state of the model world can be captured only by a full list of all agents' expectations about all possible states and corresponding actions. The convergence issue should be viewed from the perspective of the whole economy; as such, it is complex and out of the scope of this article.

3.2. Fuzzy representation of the Q-function

Since the Q-learning algorithm needs to store an estimate $\hat{Q}_t(a, s)$ for each state/action pair, state and action spaces need to be discrete and enumerable. Moreover, when state and action spaces contain many elements, the Q-learning algorithm needs to store/update a considerable number of values and becomes inefficient.
On the other hand, with only a few possible values of state variable P_t, the economy may not be able to reach market clearing equilibrium. Therefore, agents in our model cannot rely just on plain Q-learning, and it is necessary to use some variant/alteration of the original algorithm, which can operate on continuous state space.
However, it is not unreasonable to assume that households can supply either 0 or 1 units of labor, and firms can hire only an integer number of households with some upper limit; therefore, action space can be discrete.

3.2.1. Fuzzy Q-iteration

Busoniu et al. (2010) present an algorithm called fuzzy Q-iteration, which approximates the Q-function by representing it using fuzzy partition of the state space $\mathcal{S}$ and discretization of the action space $\mathcal{A}$. Since we have explained why the action space can be assumed to be discrete, we can omit the action space discretization part of the algorithm and utilize only the fuzzy partition of the state space. Using fuzzy Q-iteration instead of plain Q-learning will allow for P_t, B_t and F_t to be continuous. However, it is important to note that the algorithm is altered for our purpose. We utilize only the Q-function approximation portion. The updating process of the approximator will be different.

Let the state space $\mathcal{S}$ be divided into N so-called fuzzy sets, each described by a membership function (MF). The MFs tell us to what degree state s belongs to a particular fuzzy set. The i-th membership function $\mu_i : \mathcal{S} \to [0,1]$ has a maximum in single point $s_i \in \mathcal{S}$. The state s_i is called the core of i-th membership function. Several variants of different membership functions can be used for fuzzy Q-learning based algorithms. We will use the simplest triangularly shaped MFs.

Let the state s have D dimensions. In our case, given the state being compromised of P_t, B_t and Q_t, we have three dimensions, i.e. $D = 3$. For each dimension d of the state variable we have N_d cores $c_{d,1}, ..., c_{d,N_d}$ which satisfy $c_{d,1} < c_{d,2} < ... < c_{d,N_d}$. The triangular MFs are for each dimension defined as

$$\phi_{d,1}(s_d) = \max(0, \frac{c_{d,2} - s_d}{c_{d,2} - c_{d,1}}) \tag{22}$$

$$\phi_{d,i}(s_d) = \max[0, \min(\frac{s_d - c_{d,i-1}}{c_{d,i} - c_{d,i-1}}, \frac{c_{d,i+1} - s_d}{c_{d,i+1} - c_{d,i}})], \forall i \in \{2, ..., N_d - 1\} \tag{23}$$

$$\phi_{d,N_d}(s_d) = \max(0, \frac{s_d - c_{d,N_d-1}}{c_{d,N_d} - c_{d,N_d-1}}) \tag{24}$$

Since $P_t \in (0,1]$ and $F_t \in (0,1]$, this means that $c_{d,1} = 0$ and $c_{d,N_d} = 1$ for $d \in \{1,3\}$ and for budget dimension B_t we have $c_{d,1} = 0$ for $d = 2$ and the uppermost core c_{d,N_d} is set to arbitrary finite value - larger than B_t reaches during the course of the simulation. By taking a product of each combination of single-dimensional MFs, we get $N = N_1 * N_2 * ... * N_D$ pyramid-shaped D-dimensional MFs. [4] Let M be the number of discrete actions (or combinations of action distinct variables values if action is multidimensional - like it is the case for households) and j discrete action index, i.e. $j = 1, ..., M$. If we store parameter value $\theta_{i,j}$ for each D-dimensional MF and discrete action combination, we can define the estimate $\hat{Q}(a, s)$ over continuous state space as:

$$\hat{Q}(s, a) = \sum_{i=1}^{N} \phi_i(s)\theta_{i,j} \tag{25}$$

The basic idea of the algorithm is then updating parameter matrix $\boldsymbol{\theta}$ based on observer rewards and Bellman's principle of optimality. Each period, the agent selects an action to execute, observes the state change from s to s', receives reward $R_a(s, s')$ and updates $\boldsymbol{\theta}$. Action selection is the same as with plain Q-learning, i.e., the agent selects the action with the highest associated $\hat{Q}(a, s)$, whose index in this case is:

$$j^* = \arg\max_j \sum_{i=1}^{N} \phi_i(x)\theta_{i,j} \tag{26}$$

So instead of storing the full table of $\hat{Q}(s, a)$ values, we now have to store just the parameter vector $\boldsymbol{\theta}$ and we use the MFs to approximate the Q-function over the entire continuous state/action space. The MFs, together with the vector $\boldsymbol{\theta}$ summarize the agent's policy

[4] For more details on fuzzy representation of the *Q-function* using triangular membership functions see Busoniu et al. (2010) section 4.2.

because they are sufficient to assign an action to each state. $\boldsymbol{\theta}$ is also a source of agent heterogeneity in the simulation because it will be stored and updated for each agent separately.

In the original Fuzzy Q-iteration algorithm as presented in Busoniu et al. (2010), the state space is during the learning process visited only around the cores $c_{i,d}$. Updating process of $\boldsymbol{\theta}$ is not performed in any other states. Since we need the updating rule to be defined for all $s \in \mathcal{S}$, here, our variant of the algorithm needs to be different.

3.2.2. Updating rule

The logic of the updating rule is derived from the fuzzy Q-learning algorithm, which is described, for example, in Glowaty (2005). Let s be the current state observed by the agent, $\hat{Q}(a, s)$ the current estimate, and α learning rate. If a is the executed action and s' the resulting state, then we define δ as

$$\delta = R_a(s, s') + \beta \max_j \sum_{i=1}^{N} \phi_i(s')\theta_{i,j} - \hat{Q}_t(s, a) \tag{27}$$

where $R_a(s, s')$ is the reward received and β is discount factor. If j^* is the index of the executed action a, parameter vector $\boldsymbol{\theta}$ is updated according to the following rule:

$$\theta_{k,j^*} \leftarrow \theta_{k,j^*} + l\phi_k(s)\delta; \forall k = 1, ..., N \tag{28}$$

The updating process goes through all elements of $\boldsymbol{\theta}$ and adjusts them. The degree of change is specified not only by the learning rate α, but also by the degree of the membership of the current state s to the corresponding fuzzy set. Those elements of $\boldsymbol{\theta}$ whose corresponding MF evaluates to 0 are kept constant. On the other hand, if s is identical to one of the combinations of single-dimensional cores, the corresponding element of $\boldsymbol{\theta}$ will be changed to a full degree specified by α. However, it is necessary to note that for all other states s, i.e., states that are not identical to some combination of cores, the degree of actual change of the estimate $\hat{Q}(s, a)$ is lower than δ. This results in agents learning to be generally slower. An alternative updating rule, which addresses this issue, is presented in Appendix B.

4. Simulation

Simulation, as outlined in Algorithm 2.3.1 together with agents' decision making mechanism, was coded in Matlab 2022a[5]. Various sets of parameters, such as the number of firms and households, technology, and utility function parameters, were tested during multiple pilot runs. A graphical user interface was designed to enable the modeler to test various parameters more easily and *play* with the simulation.

The goal was to resolve whether the simulation as a whole can reach some steady state in which firms are able to create profit and households obtain utility. As stated previously, Q-learning can be used to govern an agent's behavior in a way that maximizes a function of reward. A complex simulation with many interacting agents like this one may be a very

[5]Functions mdesg_p, ndi2lin, and lin2ndi used for evaluating MFs made available online by Busoniu (2007) were utilized.

different case. Therefore, the first step was verifying whether households would even participate in the labor market and whether firms would produce and sell goods.
The second goal was to examine how a simulation that has already reached a steady state will behave when we introduce an exogenous monetary shock.
For this purpose, several aggregate economic variables were being calculated and stored during each run as the simulation progressed. Firstly, the price level P_t, which is also a state variable set to the value at which the goods market clears.[6] Consumption level was calculated as the total amount of goods consumed by the households each period. The labor participation rate was calculated as a percentage of households, which offered their one unit of labor at the labor market, i.e., chose an action for which $L_t = 1$. The portion of households that offered their labor at the labor market but were not hired by any firm is the unemployment rate. Lastly, the sum of budgets B_t of all firms and households is the money supply.

4.1. Results

Running the simulation with various sets of parameters verified that it can reach a stable state. To keep the length of the article within reason, in this section, we will look in more detail at the simulation results with only one of such sets. Nevertheless, the simulation design allows for various parameter settings, especially regarding production function and the number of households and firms.

Figure 1 presets plots of mean values of aggregate economic variables for 2500 periods of 100 simulations run in parallel with the same parameters, which were set according to the Table 1.

[6]For more details see Appendix A.

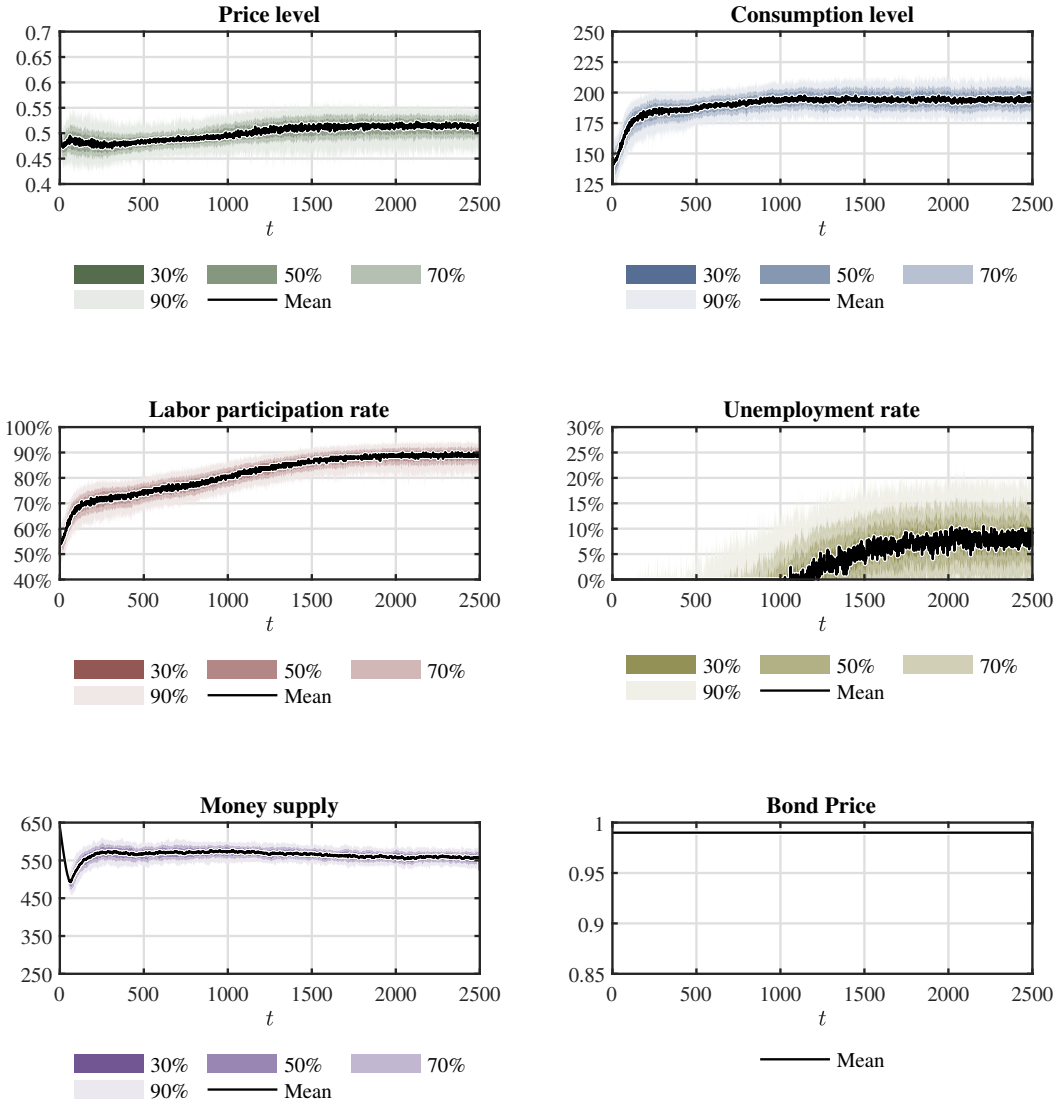

Figure 1: Simulation results

Table 1: Simulation parameters

Parameter	**value**
N_h - Number of households	120
N_f - Number of firms	100
β - Discount factor	0.95
Cl - Cost of living	0.1
Household	
σ - Utility function parameter	0.2
φ - Utility function parameter	14
I_{e_h} - Initial money holdings	2
Firm	
A - Production technology	2
α - Production function parameter	0.1
I_{e_f} - Initial money holdings	4
Q-learning	
l - Learning rate	0.9
Cores of membership functions	
- along P_t dimension of the state space	$\{0, 0.2, 0.4, 0.6, 0.8, 1\}$
- along F dimension of the state space	$\{0.8, 0.9, 1\}$
- along B_t dimension of the state space	$\{0, 1, 2, 3, 4, 5, 6, 10, 100\}$
$\mathcal{A}$ - Action space	
- c_t- Portion of housholds budget spend	$\{0, 0.25, 0.5, 0.75, 1\}$
- L_t- Housholds labor supplied	$\{0, 1\}$
- L_t- Firms labor hired	$\{0, 1, 2, 3, 4\}$
ϵ -Random action selection probability	0.1
Additional parameters	
Firm reset probability	0.1
Dividend threshold	5
Dividend ration	0.1

Simulation results are presented as mean values of multiple runs to be more consistent, repeatable and to remove the effects of random decision making.[7] To illustrate the variance of economic aggregates among simulation runs, plots include confidence intervals.
The first plot shows the price level, which after the initial span of approximately 100 periods, stabilizes under 0.5 and then gently sweeps up to stabilize just above 0.5. During the pilot runs and tuning of the simulation parameters, it has been confirmed that the price level is affected strongly by the firms' production function parameters, i.e., technology and α. The relationship between price level and technology is inverse, which aligns with intuition.
Consumption level and labor participation rate both follow a similar path. The initial steep climb of both variables is a period during which households learn that supplying labor increases their budget and enables them to increase consumption and gain utility. Households supplying labor cause production to increase accordingly. After the initial steep climb, the

[7]Each round an agent may take a random action with probability $\epsilon \in (0, 1)$.

labor participation rate increase becomes more gradual. Once it reaches a threshold in which firms' demand for labor is fully saturated, the unemployment rate starts to rise. Eventually, at roughly period 2000, all variables stabilized: labor participation rate at around 90% and the unemployment rate between 5% and 10 %. The labor participation rate did not reach 100% because ϵ portion of households and firms always choose a random action.

As has been verified during the pilot runs, the unemployment rate is mainly influenced by the ratio of firms and households and the production function parameter α. Those two parameters have explicitly been set to values that lead to a low but non-zero unemployment rate.
In Appendix B, I present a variant of the Q-function approximation and updating rule, which results in the labor participation rate climbing steeply right away to the stable 90% level, but at the expense of adding complexity to the algorithm. As shown in this example simulation, the more traditional updating rule of equation 28 results in a more gradual increase from 75% to 90%.
The money supply adjusts to its stable level once firms determine the optimal level of capital. The main factor influencing the money supply at the very beginning is initial money holdings. However, the stable level is determined by the price of bonds F and, therefore, the opportunity cost of capital.
The last plot in the figure 1 shows that F has been kept at a constant level of 0.99. F is an exogenous variable, and in the remainder of this section, we will look at how the simulation reacts to a sudden change in F, i.e., an exogenous monetary shock.

4.1.1. Monetary shock

Figure 2 shows how the simulation reacts when, at period 2500, the bond price changes from 0.99 to 0.90. The period when the exogenous monetary shock occurred is highlighted by the red dashed line.

The most noticeable is the adjustment to a new stable level of the unemployment rate and money supply.

The most noticeable are the adjustments of the unemployment rate and the price of goods to a new stable level. The unemployment rate decrease is due to capital being comparatively more costly than labor after the shock. Therefore firms hire more labor and hold less capital. The new stable price level is slightly higher, as producing one unit of production good is more costly than before.
A decrease in bond price is identical to an increase in interest rates. In a real economy, increasing interest rates should not lead to an increase in goods prices, but in this simulation, the effect is the opposite. The simplicity of the model causes this behavior. The simulation does not account for investment in any way and also lacks intermediate goods producers. Extending the model to account for those factors might alter the results.
Consumption and labor participation remain after the shock at similar levels. What is significant, however, is the slight hump in price and consumption levels that occurs during the first approximately 25 periods after the shock. For a short period, the consumption is below the new stable level, and the price level is above its new stable value.

If we look at what happens to the simulation when we introduce a reverse shock, i.e. exogenous change of bond price from 0.90 to 0.99, it becomes even more noticeable. Figure

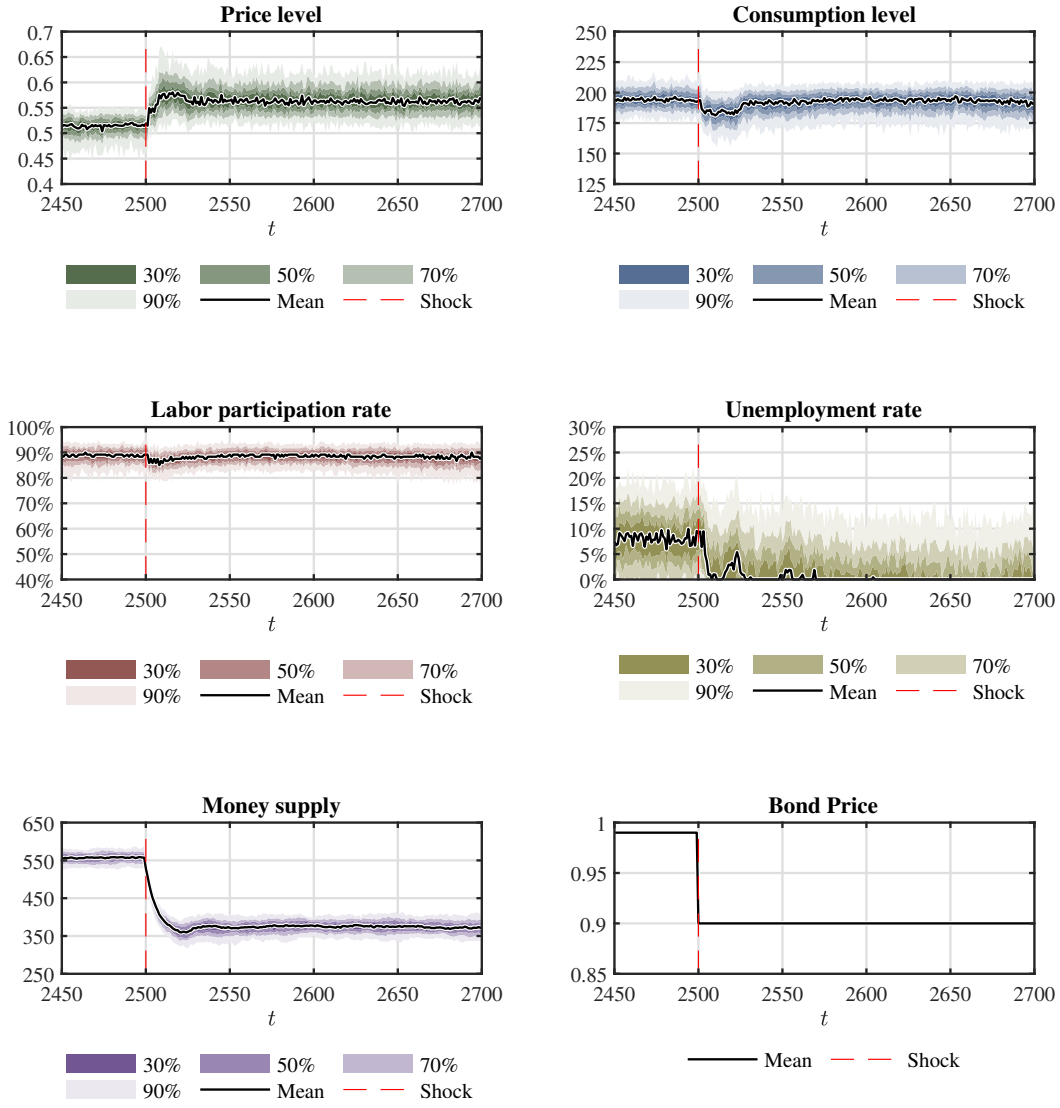

Figure 2: Monetary shock - decrease in bond price

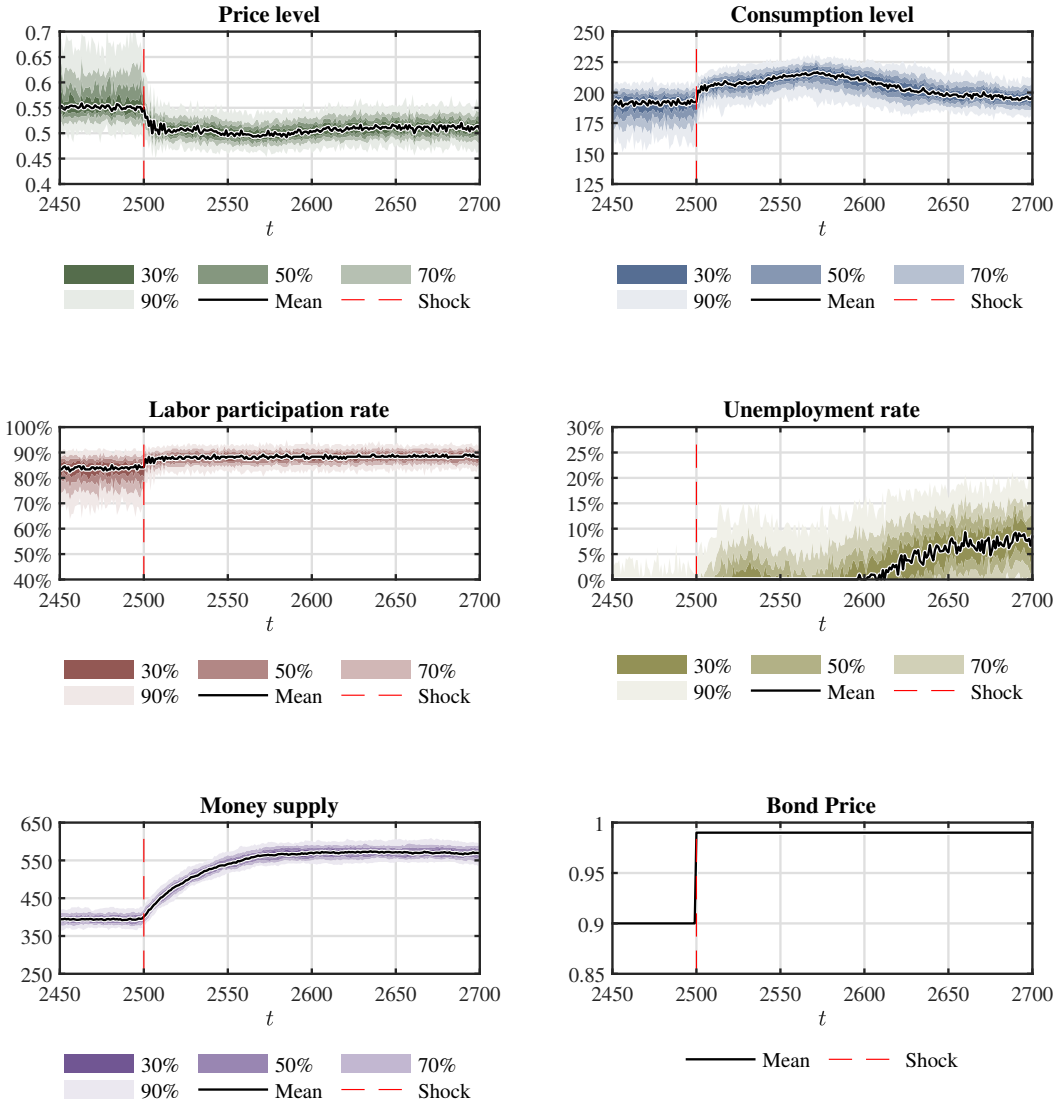

Figure 3: Monetary shock - increase in bond price

3 shows result of a change of bond price to 0.99 for a simulation that ran for the initial 2500 periods with the bond price set to 0.90.[8] By looking at the first simulation, we know that if the bond price is 0.99, then the stable level of unemployment rate should be between 5% and 10%. But after the shock we see, that it takes around 100 periods before unemployment rate starts to climb and then another 100 periods to reach stable level. During this time, consumption temporarily increases above stable level. For households, it is always beneficial to work, but if the price of bonds increases, firms should take an advantage of lover opportunity costs of capital and hire less labor. It takes a while to adjust to the new conditions as agents update their estimates of the Q-functions every period and learn the new policy. This is further slowed down by the fact, that if one agent does not follow optimal policy, it effectively creates a disturbance slowing down the adjustment of the whole simulation.
The reason why the number of periods of consumption being below stable level lower for the first shock(decrease of bond price) that the number of periods consumption was above stable level for the second shock(increase of bond price) is, that 0.90 is identical to one of the MF cores along the bond price dimension of the state. If one, or more state variable values hit one of the cores, agents are able to learn optimal action for that particular state faster.(see Appendix B)

5. Conclusion

It is indeed possible to design a computational simulation or, in other words, an agent-based model around *textboook* RBC model setting in which a reinforcement learning algorithm drives agents' decision making mechanism.
After the initial learning period, firms can produce goods, and households obtain utility. Examining the development of values of aggregate economic variables shows that the simulation can reach a steady state.
The simulation has reasonable properties, such as an inverse relationship between production technology and price level; increasing the cost of capital leads firms to substitute capital for labor and vice versa, and increasing the price of bonds leads to an increase in the overall money supply.
When agents' decision making is driven by reinforcement learning, Q-learning with a fuzzy approximation of the Q-function in this case, rigidities, and agents heterogeneity will arise in the model. A monetary shock in the form of an increase in bond price can lead to a consumption level temporally increasing above its stable level. Similarly, monetary shock in the opposite direction leads to a temporary decrease in consumption below its new stable level.
We can conclude that when using the price level, bond price, and budget as state variables, the reinforcement learning driven decision making mechanism led to the non-neutrality of monetary policy. Even when all other possible parameters of the simulation were brought over from a typical RBC model.
The structure of the simulation allows for extensibility. For example, it would be possible to add more control variables, such as investment, more types of agents, such as intermediate goods producers, or a mechanism by which new firms and households could emerge. The

[8] All the other parameters were identical.

complete source code of the simulation, including GUI, is available in the author's GitHub repository.

Appendix A.

Each period t, the market clearing mechanism sets the price of goods P_t, so that the difference between the number of goods produced by firms and consumed by households is minimized. The market clearing price is set at the intersection of the aggregate supply and demand curves. Supply and demand curves are constructed using continuous piecewise linear approximation. Let $d_0 < ... < d_N$ be non-decreasing sequence of N discretization points P dimension of the state. Since $P_t \in (0, 1]$ then we have $d_i \in (0, 1]; \forall i \in \{1, ..., N\}$. To construct an individual household's demand curve, we will evaluate the number of goods demanded as if the price were set at each discretization point. The number of goods demanded by a household h is given by

$$C_{t,h} = \frac{c_t(M_{t-1,h} + W_{t-1}L_{t-1,h} + D_{t-1})}{d_i}$$

where $W_{t-1} = 1$ and $M_{t-1,h} + W_{t-1}L_{t-1,h} + D_{t-1}$ is households budget, which is at the time when action selection mechanism is executed constant. The action selection rule gives the portion of the budget spent c_t. With probability $1 - \epsilon$, the action with the highest associated $\hat{Q}(d_i, a)$ estimate is selected, and random action is selected otherwise.

$$j^* = \arg\max_j \sum_{i=1}^{N} \phi_i(x)\theta_{i,j}$$

We get a vector of goods demanded $\vec{C}_{t,h}$ by evaluating the action selection rule at each discretization point. We can get to a piecewise linear approximation of the demand curve by summing the vectors over the complete set of households. The supply curve is constructed analogically to the demand. Evaluating the action selection rule for each firm at each discretization point will give us a vector of labor demanded $\vec{L}_f$ and a vector of capital used $\vec{K}_f$. We get the aggregate supply function by evaluating the production function for each firm and then summing the vectors of goods produced $\vec{Y}_f$ over the complete set of firms. The price P_t is finally set by finding the closest intersect of the supply and demand to the previous price P_{t-1}. In case there are no intersects of supply and demand functions, the P_t is set to 1 if $Y_{ag,i} < C_{ag,i}; \forall i \in \{1, ..., N\}$, where ag subscript stands for aggregate, and to 0.001 if $Y_{ag,i} > C_{ag,i}; \forall i \in \{1, ..., N\}$.

Appendix B.

The alternate $\boldsymbol{\theta}$ updating rule is designed as follows.
Let s be the current state observed by the agent and k_d index of its closest core along dimensions $d = 1, .., D$ (excluding $c_{d,1}$ and c_{d,N_d}):

$$k_d = \arg\min_i ||s - c_{d,i}||; i \in \{2, ..., N_d - 1\} \tag{B.1}$$

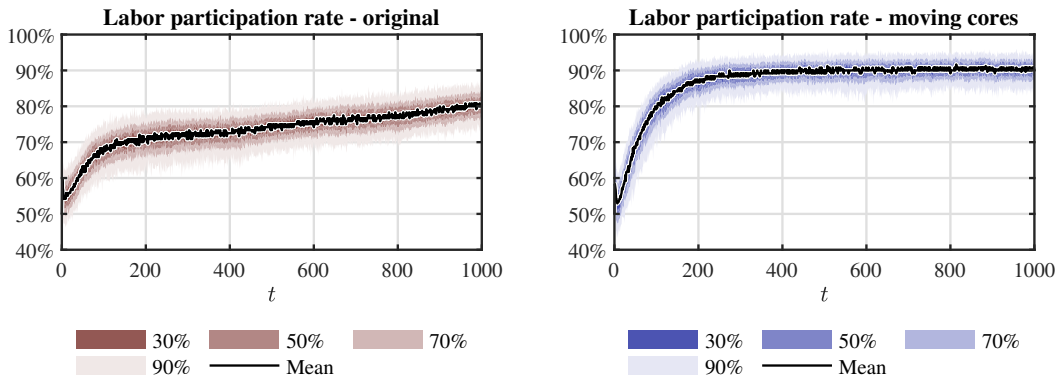

Figure B.4: Labor participation rate - comparison of updating rules

If j^* is the index of the executed action a, s' is the next state and $R_a(s, s')$ is the reward received, parameter vector $\boldsymbol{\theta}$ is updated according to the following rule:

$$\theta_{k,j^*} \leftarrow (1-\alpha)(\sum_{i=1}^{N} \phi_i(s)\theta_{i,j^*}) + \alpha(R_a(s,s') + \beta \max_j \sum_{i=1}^{N} \phi_i(s')\theta_{i,j}) \tag{B.2}$$

where α is learning rate, β is discount factor and k is index of MF, which has its maximum in point $[c_{1,k_1}c_{2,k_2}...c_{D,k_D}]$. The last step is the modification of the position of the closest core to the state s long each dimension:

$$c_{d,k_d} \leftarrow s_d; \forall d = i, ..., D \tag{B.3}$$

Simply put, the updating process sets the value of k-th element of vector $\boldsymbol{\theta}$ according to the equation B.2, and then shifts the corresponding core, so it is identical to the current state.

Figure B.4 illustrates the difference between the development of labor participation rate in the original simulation and when this different updating rule with *moving* cores was used. Both times, the simulation was run with the exact parameters, set according to Table 1. The price of bonds was set to 0.99. It is clearly visible that the updating rule with *moving* cores leads to faster convergence of the simulation to a stable state, in which the labor participation rate reaches approximately 90%.

Nomenclature

α	Production function parameter
β	Discount factor
$\hat{Q}$	Q-function estimate
$\mathcal{A}$	Set of actions
$\mathcal{M}$	Set of policies
$\mathcal{S}$	Set of states
μ	General membership function
$\phi_{d,i}$	i-th triangular membership function along d-th dimension
ϕ_i	D-dimensional pyramid shaped membership function
π	Policy

σ	Utility function parameter	I_{e_h}	Households initial money holdings
θ	Parameter of the fuzzy approximation of the Q-function	K_t	Production capital
φ	Utility function parameter	l	Learning rate
A	Production technology	L_t	Labor
a	Action	M_t	Quantity of bonds
B_t	Budget	P_t	Price of goods
C_t	Consumption	Q	Q-function
c_t	Portion of households budget spend on consumption	R	Reward
Cl	Cost of living	s	State
D	Dimension count of the membership function	U	Utility
D_t	Dividends	V	Value function of Dynamic programming
F_t	Price of bonds	W_t	Wage
I_{e_f}	Firms initial money holdings	Y_t	Production output

Code availability

The source code of the simulation presented in this article is available on author's GitHub page https://github.com/Borivoj/computational-economic-simulation or upon request.

References

Aumann, R.J., 1964. Markets with a continuum of traders. Econometrica 32, 39–50. doi:10.2307/1913732.

Bellman, R., 2010. Dynamic Programming. Princeton University Press.

Bernanke, B., Gertler, M., Gilchrist, S., 1998. The Financial Accelerator in a Quantitative Business Cycle Framework. Working Paper 6455. National Bureau of Economic Research. doi:10.3386/w6455.

Busoniu, L., 2007. Approximate rl and dp toolbox. URL: https://busoniu.net/repository.php.

Busoniu, L., Babuška, R., Schutter, B.D., Ernst, D., 2010. Reinforcement learning and dynamic programming using function approximators. CRC Press.

Calvo, G.A., 1983. Staggered prices in a utility-maximizing framework. Journal of Monetary Economics 12, 383–398. doi:10.1016/0304-3932(83)90060-0.

Christiano, L.J., Eichenbaum, M.S., Trabandt, M., 2018. On dsge models. Journal of Economic Perspectives 3, 113–140. doi:10.1257/jep.32.3.113.

De Grauwe, P., 2008. DSGE-Modelling: when agents are imperfectly informed. Working Paper Series 897. European Central Bank. doi:10.2139/ssrn.1120763.

Dosi, G., Napoletano, M., Roventini, A., Stiglitz, J.E., Treibich, T., 2020. Rational heuristics? expectations and behaviors in evolving economies with heterogeneous interacting agents. Economic Inquiry 58, 1487–1516. doi:10.1111/ecin.12897.

Fagiolo, G., Roventini, A., 2017. Macroeconomic policy in dsge and agent-based models redux: New developments and challenges ahead. Journal of Artificial Societies and Social Simulation 20, 1. doi:10.18564/jasss.3280.

Galí, J., 2008. Monetary Policy, Inflation, and the Business Cycle: An Introduction to the New Keynesian Framework. Princeton University Press.

Gertler, M., Kiyotaki, N., 2015. Banking, liquidity, and bank runs in an infinite horizon economy. American Economic Review 105, 2011–43. doi:10.1257/aer.20130665.

Glowaty, G., 2005. Enhancements of fuzzy q-learning algorithm. Computer Science 7, 77–77. doi:10.7494/csci.2005.7.4.77.

Jump, R.C., Hommes, C., Levine, P., 2019. Learning, heterogeneity, and complexity in the new keynesian model. Journal of Economic Behavior & Organization 166, 446–470. doi:10.1016/j.jebo.2019.07.014.

Kiyotaki, N., Wright, R., 1989. On money as a medium of exchange. Journal of Political Economy 97, 927–954. doi:10.1086/261634.

Kollman, R., 1996. The Exchange Rate in a Dynamic-Optimizing Current Account Model with Nominal Rigidities: a Quantitative Investigation. Cahiers de recherche 9614. Universite de Montreal, Departement de sciences economiques. URL: https://ideas.repec.org/p/mtl/montde/9614.html.

Mankiw, N.G., Reis, R., 2007. Sticky Information in General Equilibrium. Journal of the European Economic Association 5, 603–613. doi:10.1162/jeea.2007.5.2-3.603.

Marimon, R., McGrattan, E., Sargent, T.J., 1990. Money as a medium of exchange in an economy with artificially intelligent agents. Journal of Economic Dynamics and Control 14, 329–373. doi:10.1016/0165-1889(90)90025-C. special Issue on Computer Science and Economics.

Mitchell, T.M., 1997. Machine Learning. 1 ed., McGraw-Hill.

Sinitskaya, E., Tesfatsion, L., 2015. Macroeconomies as constructively rational games. Journal of Economic Dynamics and Control 61, 152–182. doi:10.1016/j.jedc.2015.09.011.

Smets, F., Wouters, R., 2003. An estimated dynamic stochastic general equilibrium model of the euro area. Journal of the European Economic Association 1, 1123–1175. doi:10.1162/154247603770383415.

Sonnenschein, H., 1972. Market excess demand functions. Econometrica 40, 549–563. doi:10.2307/1913184.

Tesfatsion, L., 2017. Modeling economic systems as locally-constructive sequential games. Journal of Economic Methodology 24, 384–409. doi:10.1080/1350178X.2017.1382068.

Tversky, A., Kahneman, D., 1974. Judgment under uncertainty: Heuristics and biases. Science 185, 1124–1131. doi:10.1126/science.185.4157.1124.

Watkins, C.I., Dayan, P., 1992. Technical note - q-learning. Machine Learning 8, 279–292. doi:10.1023/A:1022676722315.