Simulation of the Economy: A Deep Reinforcement Learning Agent-Based Model Approach

Daniel D. Ortega

Northern Illinois University

Advisor: Dr. Evan Anderson

May 8, 2021

**Abstract**

This paper will be containing a prosumer utility-leisure maximization problem within a Markov Decision Process (MDP) framework. There will be seven continuous state variables, which are nominal price, wealth, debt, capital, interest, labor, and inventory, and 15 discrete actions, which will fall into six types: selling inventory, crafting, borrowing money, paying back debt, purchasing capital, and consuming. A single deep reinforcement learning (RL) agent will be utilized to find an optimal policy within this framework. This framework allows for incorporation of incomplete information within the state variables, such as nominal price state include exogeneous shocks such as inflation and demand shocks, which are not directly known to the agent. In this paper, I will analyze this agent-based model (AMB) an demonstrate how the strategies of an adaptive agent, under certain market conditions, influence macroeconomic variables such as output, savings, and consumption. I will establish a baseline economy, and will be testing many hypotheses such as how will an agent react with or without market power? Or how will the agent react under contractionary or expansionary monetary policies?

Table of Contents

[Introduction 4](#_Toc71469430)

[Related Work 6](#_Toc71469431)

[Markov Decision Process 8](#_Toc71469432)

[Deep Q Learning 10](#_Toc71469433)

[Model 13](#_Toc71469434)

[Action space 14](#_Toc71469435)

[State Space 16](#_Toc71469436)

[Transition Function 18](#_Toc71469437)

[Reward Function 19](#_Toc71469438)

[Q Network 20](#_Toc71469439)

[Results 21](#_Toc71469440)

[Conclusion 22](#_Toc71469441)

[References 23](#_Toc71469442)

# Introduction

In the 1970s, there was a paradigm shift in economics thanks to what is known as the Lucas Critique. The main argument of the critique was that it was foolish to try to explain macroeconomic phenomenon using historical time series data because such econometric models lack any theoretical foundations, and the parameters quickly become invalid under policy changes (Lucas 1976). This critique has led to the wide usage of Dynamic Stochastic General Equilibrium in macroeconomics, which are a step in the right direction, as they incorporate micro foundations, which are the decisions of an agent such as consumer or firm. These economic agents are assumed to be rational which means they know their preferences and will act accordingly to maximize their consumption in an optimal manner. This underlying assumption makes optimization problems much easier, and it gives internal consistency to many macroeconomic models via rational expectations. However, these are strong assumptions which downplay the importance of independent agents and simplifies their actions for mathematical convenience. As pointed out in Tesfatsion and Judd (1996), many general equilibrium market clearing conditions are brought about by generalizing the actions of agents via mechanisms such as the Auctioneer in Walrasian equilibrium.

A more dynamic model would be one in which agents are independent, and thanks to reinforcement learning techniques, this is possible by allowing an agent to learn by interacting with its environment. This replaces the common assumption of rationality for the assumption of bounded rationality, which implies that an agent’s rationality in decision making is limited by many variables such as information and time; agent must instead learn to adapt to market forces. (Simon 1990). These agents have been coined as Artificial Adaptive Agents (AAA) and hold several modeling advantages over traditional methods. Since agents learn strategic behaviors endogenously within this framework, this allows for explicit analysis on the strategies of the agent as well as the ability to simulate controlled experiments (Holland and Miller 1991). Simulations can be carried out in which the environment the agent interacts with can be defined as a Markov Decision Processes (MDP). MDPs are commonly used to express optimal control problems in economic, however, most economic optimal control problems are found to have less than two continuous state variables, as any more makes the problem exponentially harder.

The agent in the model presented in this paper, will be represented as both the consumer and producer, a common tactic in many economic models such as the Lucas Island Model or Robinson Crusoe Model. There will be seven continuous state variables which will represent the dynamics of the economy: nominal price wealth , debt , capital , interest rate , labor , and inventory . There will be 15 discrete actions, which will fall into six action types: selling inventory, crafting, borrowing money, paying back debt, purchasing capital, and consuming. This discretizing of six action types into two discrete levels makes the problem much easier, as Q learning maximizes the control problem over the set of possible actions. Such a highly dimensional problems is intractable and not possible to solve using analytical methods and requires a numerical approach. Therefore, I will be using deep reinforcement learning methods, specifically Deep Q Learning, to compute an optimal policy, as it one of the only methods to be able to handle multiple continuous state variables (Mnih V., 2013).

One drawback of this technique is that it will only be able to approximate the optimal policy, unlike other methods such as dynamic programming which guarantee optimal convergence. Meaning agents can get easily stuck in local optimum, which is still a matter worth studying as ACE modeling emphasizes heterogeneous agents due to bounded rationality. A suboptimal policy can be deemed satisfactory for an agent with bounded rationality, as such policies are still complex and are possible strategies a firm or consumer might take in real life. Fortunately, Deep Q Learning has had many advances in heuristics that allow a better probability of converging to an optimal solution such as experience replay, recurrent networks, and double Q networks.

Ultimately, the goal of the agent will be to maximize its consumption using it consume action. The agent will receive a large reward when it consumes and to consume the most it possibly can, the agent needs to earn income. To do that the agent must sell its goods on the market while trying to predict the changes in nominal price that are due to changes in demand. The agent will wants to sell its goods at the highest price, but only when the price change is due to an increase in demand. Meaning if there is an increase in price due to factors such as the monetary shock, and the agent believes the changes was due to an increase (decrease) in demand and it will produce more (less), the consequences for such wrong predictions will be an oversupplying (undersupplying) of the market. This postulate of incomplete information in prices and the idea that the agent must try and predict demand shocks amongst monetary noise was first formulated by Lucas (1973), which helped explain the short-term relationship between unemployment and inflation. The agent will have the ability to borrow money, paying a fixed amount of interest each time step. The agent will also need to invest in capital, as it depreciates each time step and is important factor in production. I will establish a baseline economy, and will analyze the agents behavior under different monetary policies as well as under monopoly and competitive market conditions.

# Related Work

When it comes to the application of Deep Q learning, a sub technique of Deep Reinforcement Learning, there are many examples of the technique producing human level control in many environments such as the game Go (Mnih V., 2015). Recently within the field of economics, Deep learning techniques have been utilized in many ways in the field of economics, from creating auctioneer policies to predicting stock prices (Mosavi A., 2020). The combination of an adaptive agent via Deep Q learning and the environment provided by the MDP is a type of agent-based model. Agent-based models themselves are prevalent in the economic literature, with their computational counterpart (ACE) becoming more prevalent. I will be covering the examples which fall under the ACE paradigm that attempt to model macroeconomic phenomena using deep RL agents.

A recent example done in (Chen M., 2021), where the agent had three continuous actions which were consumption, bond saving, and hours worked. There are seven continuous state variables which include last period’s money supply, bond holdings, inflation, consumption, hours worked, as well as monetary and technology shocks. The agents’ actions are only able to interact with three of the seven states, which are inflation, real consumption, and real bond holdings. The environment for the agent is based off a dynamic stochastic general equilibrium (DGSE), and the agent was found to have converged to the two steady states featured in that model. This simulation is very similar in essence to my model, where RL agents are utilized to see how their strategies interact with a continuous-state model of the economy. The main difference is that there are only three action types, which are continous, and only interact with three of the seven state variables. In my model there are six action types which interact with all the state variables, allowing for a more dynamic model. Since the action and states are continuous, this model utilizes an actor critic algorithm versus a Deep Q Learning algorithm. It also incorporates the monetary shocks within the state space, unlike my model in which the monetary shocks are only partially observable via the nominal price.

Another recent example which utilizes Deep RL agents in a simulation of an economy is in (Zheng S., 2020). In their model the environment is grid-based, where the RL agents can move and collect resources such as wood and stone. These resources are then used to allow the agent to build a house on the grid, which will reward them with a certain amount of coin depending on their building skill. This building skill is predetermined and is meant to be a representation of how much income or coin is gained per unit of labor. Agents are then able to specialize in collecting resources, selling resources, and building houses. The authors use this model as a bench line to determine optimal taxation via another deep RL agent who acts a central income tax planner. Their simulation demonstrated complex behavior by all the agents, and the AI tax policy was optimal based off an equality-productivity criterion. This is a successful example on how to do a controlled, simulated experiment in economics. While the environment is grid-based, it captures many of the same dynamics as my model, such as the tradeoff between labor and income. It also has the advantage of introducing multiple heterogeneous autonomous agents who learn differing strategies.

# Markov Decision Process

A Markov decision process is a modeling framework introduced by Bellman (1957), which represents dynamic decision-making done by an agent in an environment, and is described as the 4 tuples (S, A, P, R). Where S is the set of states and A is the set of actions (available in each state), P is the transition probabilities and R is the reward received from transitions. Under such a framework, an agent starts in an initial state, takes an action, and then transitions to a new state, receiving a scalar reward. Each transition is equal to one time step and therefore the framework can be expressed as either: (1) A finite horizon model in which the number of time steps is limited, and the total reward is defined as or (2) an infinite horizon model in which there the steps approach infinity, and the total reward includes a discount factor and is defined as , where the discount factor is between zero and one and T is the total time steps. The goal in such a framework is to find an optimal policy, which will tell the agent the best action taken given any state.

Such a technique is widely utilized in economics, as the ability define the value of a transition sequence with a scalar reward allows for the use of optimization techniques to find the optimal policy. The optimal policy satisfies the bellman equation:

Where is the current available action, r(s,a) is the initial reward and the second term is the discounted value of all future rewards. Where is the discount value, V is the value function of a given state, and T is the current transition to a new state given the current action and state. While an optimal policy will follow this optimality rule, the main optimization problem is the reward function itself. Which for economics, is commonly the maximization of consumption for consumers and the maximization of profits for producers.

Compared to more explicit models commonly found in economic optimal control problems, the state transitions will follow a generative approach. Meaning rather than new transition states being sampled from an explicit probability distribution, samples are generated via a generative model, which is a single step simulator that can take any state and action and output a new state and reward (Kearns ., 2002). For example, one of the state variables is the nominal price of the good the agent is selling; this nominal price depends on random monetary and demand shocks, as well as supply shocks (for monopoly power example) which are caused by the agent when it oversupplies or undersupplies the market. All these shocks are added onto the nominal price each time step, and the simulator can generate a new price given any current price. Meaning as the simulation moves forward, the nominal price moves in a random drift with supply shock influence from the agent (monopoly example). This incorporates incomplete information, as the agent only receives the nominal price as a state input into the deep neural network.

While this problem can be described as a Partially Observable Markov Decision Process (POMDP), where an agent only sees a set of observations instead of the actual underlying states, such a distinction is only necessary for solvers which attempts to assign probabilities of the true state given an observation. These probabilities over the state space, known as the belief state, are not feasible for continous state problems. The Deep Q Network will take the states as inputs regardless of this distinction.

# Deep Q Learning

Deep Q learning is the combination of the reinforcement learning technique, Q learning (Watkins 1989), and neural networks to approximate an optimal policy. The technique was first successfully introduced by Google’s DeepMind team in which agents learned complex strategies in several Atari Games (Mnih V., 2013). Q Learning is a model free approach, meaning that the transition function is not explicitly defined for the agent and it must learn the dynamics of the model via an exploration policy. Q learning has been proven to converge to optimal policies (Watkins & Dayan 1992). It does this by assigning values to state action pairs via the Q function, called Q-values. However, with continuous state variables, storing these values in a Q table becomes quickly intractable. Therefore, a neural network, a universal function approximator, is utilized to approximate the Q function. The Q function is defined as:

where is the policy which maps sequences to actions (Mnih V., 2013). The optimal Q function will maximize the bellman equation which in this context can be expressed as an expected value:

The neural network has weights , and is called the Q network. The training of the network is done by minimizing the following loss function using stochastic gradient descent:

Where and is defined as the target Q function. The target Q function is updated periodically throughout the training session and the frequency of the update is an important factor, because if it is too low, the agent will be essentially chasing a moving target. The training process begins when the states are inputted into the Q network via forward propagation, the loss is calculated, then via stochastic gradient descent loss is minimized. Stochastic gradient descent utilizes backpropagation, which is done by calculating the gradient of the loss function w.r.t to the Q network weights by the chain rule and then iterating backwards from the final layer (Goodfellow I., 2016). P(s,a) is defined as the a probability distribution over states s and actions a is called the behavior distribution. This technique is known as off-policy learning because the agent learns a greedy strategy while assuring that there has been sufficient exploration of the state space via the behavior distribution. To determine the behavior distribution, in my model, the agent follows a SoftMax exploration algorithm, which is based off the Boltzmann distribution. Where the probability of selecting an action is expressed as . As approaches zero, the agent will choose actions that that have the highest Q values, and when , the agent selects actions randomly. (Vamplew P., 2016). This is type of exploration policy is known as a epsilon-greedy strategy and is a meant to balance the tradeoff between exploration and exploitation.

Further improvements in this algorithm include experience replay, prioritized experience replay, double Q learning, and recurrent networks. Experience replay allows for a collection of transitions to be stored in what is called the Buffer Replay (D) and was developed in (Lin 1992). This allows the agent to essentially store previous decisions made in each state, and what reward was received. The agent will be able to sample tuples (s,a,r,s') uniformly from this replay buffer D. This will allow a more random collection of recent experiences for the agent to encounter again. Allowing the agent to essentially redo a random mix of past decisions as a means of better exploring the state action space. This will allow more efficient training of the Q network, as it prevents any temporal correlation bias from only receiving the data in a sequence. Experience replays by itself only allows the storage of the most recent transitions and does not decide which sequences should have more of a priority of being replayed.

Rather than just sampling randomly from recent experiences in the buffer, prioritized experience replay, introduced in (Schaul T., 2016), allows specification on which experiences should be given the most priority when sampling. The main underlying principle being that the sequences with the highest errors have the most to learn from. Error ) being defined as the distance from the target Q value y and priority P is defined as , where is a small constant that prevents any sequence from having zero priority. This priority is then translated to a probability of being chosen to replay is defined as . Prioritized alpha is how much probability is given to choosing a sequence from the buffer with a high priority versus choosing uniformly from the buffer. When 0 means all actions are uniformly randomly sampled from the buffer and when 1 the full weight in terms of probability of occurring is given to the sequences. Meaning those sequences which have the highest errors will have the highest probability of being sampled. However, this implementation of prioritizing sequences with high errors, naturally makes the sample biased towards these sequences. To reduce this bias, importance sampling weights are added, where .

The last innovation I will be utilizing for the deep Q learning algorithm is known as Double Q Learning and was presented in (Hasselt H.V., 2015), and is meant to reduce the bias of overestimated Q values. This bias comes from the fact that the Q learning algorithm and DQN use the same Q values to choose and evaluate an action. This bias is especially present in environments which have a lot of noise, as an action is less likely to be optimal given similar states. The idea is to have two separate Q functions to reduce this correlation of selection of the best action and evaluation of the action. It does this by randomly choosing one of the Q functions for determining the greedy action while simultaneously updating the value of the action state pair using the other Q function. This essentially separates the task of choosing an action and evaluating an action between the two Q functions, breaking down this bias.

# Model

The model will be represented as a MDP with 15 discrete actions and 7 continuous state variables. The main focus will be on the predictions an agent must make about the price direction of its products, in which such decisions will decide how much quantity to supply of their product. The agent must face decisions such as: are the changes of price in my product due to changes in real demand or inflation? The agent wants to produce equal to its real demand as to sufficiently supply the market. If there is an inflation shock, such as a central bank printing more money, a firm must know how much of the increase in price is due to either a demand or inflation shock (Lucas 1973). If the firm predicts incorrectly, that will mean the agent did not maximize its profit and in the case of the monopoly power example, would mean either a under or oversupplying of the market, causing a supply shock on the nominal price state variable.

As mentioned previously, the agent will be able to invest in capital and borrow money, the former being more necessary than the latter. I predict an optimal strategy in the competitive market under either monetary policy will be a combination of crafting, selling at the mean of the monetary shock, investing in capital periodically, and ultimately consuming. As for the monopoly power example, I expect that the agent will undersupply the market, as theory suggest, and will benefit from selling its good at a very high price. The order and magnitude of such decisions is up for question, and is of particular interest, as such actions can potentially explain phenomena such as the business cycle; As there might be times where the agent is consuming small or large amounts, and times where the agent is producing small or large amounts.

## Action space

1. Sell: Sell Q amount of good. is the quantity good supplied and sold, defined as . Meaning as demand fluctuates between -2 and 2, the agent wants to sell between 1 to 5 goods. Meaning if the agent chooses to sell 5 goods, then inherently he is assuming The action type of selling illustrates how the agent must predict the change in demand based off the nominal price. Since the demand shock fluctuates between five levels, the agent’s action of selling is five levels. By choosing to sell a Q amount of good, the agent needs to make a guess at what the change in real demand is, that way he is producing correctly. Therefore, the market clearing condition is when
2. Craft: Produce items using labor and capital. There are two discrete levels of production, one that cost 15 labor units and another that cost 30 labor units. Production function is characterized as the Cobb-Douglas function , where =.5, = .5, and A = 1.
3. Invest: Invest money to buy capital. There are two discrete levels of capital investment, one that cost $100 and another that cost $200. The agent must have enough funds in the wealth state to afford either choice plus the interest payment for that step.
4. Borrow: Take on debt. There are two discrete levels of debt the agent can take on which are $100 and $200.
5. Payback: Pay back debt. Same pattern as previously, there will two discrete levels of debt-payments, $100 and $200. The agent wealth must be greater than or equal to the amount paid plus the interest payment the agent owes for the period.
6. Consume: This ultimately will be the main objective of the agent, however, to consume will not have a fixed cost as the other actions did. Rather there will be two discrete types of consumption which can be charactered in the following Keynesian consumption function: . Where a is the minimum level of consumption, b is the marginal propensity to consume, and Y are the savings of the agent. For the first level of consumption, a is 50 and b = .10. For the second level of consumption, a is 100 and b = .20. For the agent to be able to perform these actions, it must have the required wealth specified in the consumption function. I will be referring to the two levels of consume as small consume and large consume.

For all the actions except sell, there are two levels, one with a larger magnitude than the other. Therefore, the remainder of this paper I will be referring to each action as either small or large. (i.e small invest is the $100 and large invest is the $200). For the sell action, I will be numbering sell 1-4 with each level referring to how much output or inventory is sold.

## State Space

1. Nominal Price (): Defined as , where is the demand shock, is the monetary shock, and is the supply shock. Z and M are randomly sampled from normal distributions, where Z~N(0,.7) and M~(0,.6). Both are rounded to the nearest integer, so that they both range from -2 to 2. This allows the agent to guess the change in demand as four discrete actions. If the guess in demand is greater than the actual demand and the agent ends up oversupplying the market, then S = , which is the distance between the demand shock guess (which is implied by the sell action) and the actual demand shock. Meaning the more quantity sold and oversupplied, the greater magnitude in which the nominal price will decrease (which ranges from 0 to 5. This ability to oversupply the market will remain the same for both examples, the differentiating factor is the market power example, which will have the ability to undersupply. Meaning if for the market power example, then S = , giving the agent the ability to undersupply the market thereby increasing the price of the good.
2. Savings (): Defined as the following equation , where C is consumption, D is debt, i is the interest rate, is a constant to reduce the interest payment per step, which is equal to 6 meaning every 6 steps, the agent will pay the amount of the interest rate multiplied by the debt back. is the quantity good supplied and sold, defined as . Meaning as demand fluctuates, the agent can sell between 1 to 5 goods. Is zero unless the agent chooses to sell its goods.
3. Debt (): Defined as ) – PB + BR, where is the interest gained every step, C is equal to six, and is meant to reduce the amount of interest owed relative to the actual interest rate. PB is the amount paid back when the agent chooses to payback debt, either $100 or $200. BR is the amount borrowed when the agent chooses to borrow, either $100 or $200.
4. Capital (): where is the depreciation rate and is set to .95, and BT is the amount of capital bought when the agent chooses either investment actions.
5. Interest (): . This equation demonstrates how a decrease in money supply via an increase in debt results in a higher interest rate. Meaning as the agent’s demand increases for debt, so too will the interest rate. Also incorporates the impact of the money supply shock, with a positive (negative) money supply shock lowering (increasing) the interest rate.
6. Labor (): . This occurs every step and no actions needs to be taken for the agent to accrue labor. The agent can hold at max 24 labor units, representing a time constraint. Meaning every 8 steps can be interpreted as one day. KT is zero, unless the agent chooses to produce inventory,
7. Inventory ( , where y is the amount of goods produced by the agent when it uses its produce action; can range from depending on the levels of capital and labor.

## Transition Function

If action is sell Q amount, and Q and = hen and (recall Q = ). If Q and > (oversupply) then , and where S = . If < (for market power example, zero otherwise) then S = .

If action is small invest + then where . If action is large invest and + then where .

If action is small craft, then (Recall: . If action is large craft.

If action is small borrow then ) + 100. If action is large borrow and then ) + 200.

If action is small payback and ) - 100. If action is large payback and ) - 200.

To ensure the agent is following certain state constraints is specified in the transition function. If the agent attempts an action which is not possible, then the simulation will reset so the agent restarts at step 1 the initial state and receives a large negative reward. Actions that are not possible include: Spending money when there is not enough, either by investing, consuming, or paying back debt; Crafting when there is not enough labor; Or selling Q inventory when there is not enough inventory. The initial state is where is equal to {15.0, 105.0, 0.0, 5.0, .02, 12.0, 0.0}. Starting the agent off with these initial states allow for easier training, as the agent can do any of the small actions early on.

## Reward Function

If current wealth is equal to zero, then r = - otherwise reward is defined as the following function: . If the agent chooses an action which is not possible than the simulation will reset, and the agent will receive a reward of -50,000. This mechanism coupled with Prioritized Replay will ensure more efficient training, as the sequences where the agent chose an impossible action, will have a higher probability of being replayed, giving the agent another chance fix its mistakes. Ultimately the agent will be incentivized to increase the number of plausible steps, as steps increase the overall reward will increase as well. The reward equaling - when , is put in place to ensure the agent does not just take the greedy action of borrowing repeatedly. Eventually if the agent borrows too much, the interest payments will bring the wealth down to zero, giving the agent a high negative reward. Overall, this reward function intends to capture the consumption-leisure dynamics, as well as the profit maximization incentives of a firm. The idea of receiving scalar rewards via a reward function not only allows for optimization of such highly dimensional problems, but in the case of economics, represents the foundational idea of incentives. How the reward function is shaped ultimately shapes the incentives of the agent, and will be reflected in the optimal policy.

# Q Network

The Q network is a recurrent neural network with one hidden layer. The first node in the Q network utilizes a Long Short-Term Memory (LSTM) layer, which is robust against time series data (the type of data being generated in this model) and can learn long-term dependencies. The seven states are inputs to the Q network with the predicted optimal action as the output. Using a recurrent neural network with a LSTM node for a Q network has been successfully used many times to make predicts based off time series data (Hausknecht & Stone, 2015; Lipton Z.C., 2017). It does this by having three gates which have certain functions such as storing, outputting, and discarding information. The LSTM inputs the seven states and propagates the information forward into 21 outputs. The final output layer condenses these 21 nodes into 15 action nodes. It uses a SoftMax activation function, which is essentially the logistic function but with multiple dimensions. Training the Q network, allows for output layer to properly assign probabilities to each action given the states.

Due to the previously mentioned innovations in Deep Q Learning, there are many hyperparameters that need to be specified. This creates another disadvantage to Deep Q Learning, as convergence is very sensitive these hyperparameters, making tuning of these hyperparameters very time-consuming (Fernandez and Caarls, 2018). These are the hyperparameters specifications that I have found to have allowed for stabile training of this model: target update frequency is 500, Learning Rate is .05, buffer size is 300, batch size is 50, prioritized replay and are .6.

# Results

The results are showed below in these two tables. Table 1 illustrates the sars tuple for the agent within a competitive industry. It can be see that the agents’ actions converged to a strategy of small consume, small craft, and then sell one unit. As previously mentioned, when approximating the optimal policy, convergence is not necessarily guaranteed. This policy is obviously not optimal, as the agent refuses to invest in any new capital and continues this pattern until it is unable to produce any more goods.

# Conclusion

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1 |  |  |  |  |  |  |  |  |  |  |
| SARS Tuple for Competitive Market example where Inflation Shock ~N(.02, .55) | | | | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | Action | Reward |
| -1 | 0 | 15 | 105 | 0 | 5 | 0.02 | 12 | 0 | S-Consume | 3157.5 |
| -1 | 0 | 14 | 44.5742 | 0 | 4.75 | 0.02 | 15 | 0 | S-Craft | 57 |
| 0 | 1 | 13 | 44.6485 | 0 | 4.5125 | 0.02 | 6 | 7.54 | Sell Q =1 | 1 |
| 0 | 1 | 14 | 83.7879 | 0 | 4.2869 | 0.012 | 9 | 6.54 | Sell Q =1 | 1 |
| 1 | 1 | 15 | 125.917 | 0 | 4.0725 | 0.005 | 12 | 5.54 | Sell Q =1 | 1 |
| 0 | 0 | 17 | 185.988 | 0 | 3.8689 | 0 | 15 | 4.54 | S-Consume | 3934.78 |
| 0 | -1 | 17 | 117.39 | 0 | 3.6755 | 0 | 18 | 4.54 | Sell Q =1 | 1 |
| 0 | 1 | 16 | 168.39 | 0 | 3.4917 | 0.008 | 21 | 3.54 | S-Consume | 5204.13 |
| 0 | 0 | 17 | 101.616 | 0 | 3.3171 | 0 | 24 | 3.54 | S-Craft | 39.8052 |
| -1 | 1 | 17 | 101.616 | 0 | 3.1512 | 0 | 15 | 9.85 | Sell Q =1 | 1 |
| 0 | 0 | 17 | 135.616 | 0 | 2.9937 | 0 | 18 | 8.85 | Sell Q =1 | 1 |
| 0 | 0 | 17 | 186.616 | 0 | 2.844 | 0 | 21 | 7.85 | S-Consume | 5247.88 |
| 0 | 0 | 17 | 117.954 | 0 | 2.7018 | 0 | 24 | 7.85 | Sell Q =1 | 1 |
| -1 | 0 | 17 | 168.954 | 0 | 2.5667 | 0 | 24 | 6.85 | S-Consume | 5856.18 |
| 0 | -1 | 16 | 102.059 | 0 | 2.4384 | 0 | 24 | 6.85 | Sell Q =1 | 1 |
| 0 | 1 | 15 | 150.059 | 0 | 2.3165 | 0.008 | 24 | 5.85 | S-Consume | 5805.16 |
| -1 | -1 | 16 | 85.1074 | 0 | 2.2006 | 0 | 24 | 5.85 | Sell Q =1 | 1 |
| 0 | 0 | 14 | 117.107 | 0 | 2.0906 | 0.008 | 24 | 4.85 | S-Consume | 5716.19 |
| 0 | 1 | 14 | 55.4321 | 0 | 1.9861 | 0.008 | 24 | 4.85 | S-Craft | 23.8329 |
| 1 | -1 | 15 | 55.4677 | 0 | 1.8868 | 0 | 15 | 9.74 | Sell Q =1 | 1 |
| -1 | 0 | 15 | 115.468 | 0 | 1.7924 | 0.008 | 18 | 8.74 | Sell Q =1 | 1 |
| -1 | 1 | 14 | 145.561 | 0 | 1.7028 | 0.008 | 21 | 7.74 | S-Consume | 5149.35 |
| 0 | 0 | 14 | 81.0568 | 0 | 1.6177 | 0 | 24 | 7.74 | Sell Q =1 | 1 |
| 0 | -1 | 14 | 123.057 | 0 | 1.5368 | 0 | 24 | 6.74 | Sell Q =1 | 1 |
| 0 | 0 | 13 | 165.057 | 0 | 1.4599 | 0.008 | 24 | 5.74 | S-Consume | 5845.65 |
| 0 | -2 | 13 | 98.6143 | 0 | 1.3869 | 0.008 | 24 | 5.74 | Sell Q =1 | 1 |

# References

Bellman R. (1957). A Markov Decision Process. *Journal of Mathematics and Mechanics.*

Chen M., Joseph A., Kumhof M., Pan X., Shi R., Zhou X. (2021). Deep Reinforcement Learning in a Monetary Model. *Econometrics.*

Fernandez F.C., Caarls W. (2018). Parameters Tuning and optimization for Reinforcement Learning Algorithms Using Evolutionary Computing*. IEEE.*

Goodfellow I., Bengio Y., Courville A. (2016). Back-Propagation and Other Differentiation Algorithms. *Deep Learning.*

Hasselt H.V., Guez A., Silver D. (2015). Deep Reinforcement Learning with Double Q-Learning. *Machine Learning.*

Hausknecht M., Stone P. (2017). Deep Recurrent Q-learning for Partially Observable MDPs. *Machine Learning.*

Holland J.H., Miller J.H. (1991). Artificial Adaptive Agents in Economic Theory. *American Economic Association.*

Kearns M., Mansour Y., NG A. (2002). A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. *Machine Learning.*

Lucas R. (1973). Some International Evidence on Output-Inflation Trade-offs. *American Economic Review.*

Lucas (1975). An Equilibrium Model of the Business Cycle. *Journal of Political Economy.*

Mnih, V., Kavukcuoglu, K., Silver, D. et al. (2015). Human-level control through deep reinforcement learning. *Nature.*

Mnih, V., Kavukcuoglu, K., Silver, D. et al. (2013). Playing Atari with Deep Reinforcement Learning. *Machine Learning.*

Mosavi A., Faghan Y., Ghamisi P., Duan P., Ardabili S.F., Salwana E., Band S.S. (2020). Comprehensive Review of Deep Reinforcement Learning Methods and Applications in Economics. *Mathematics.*

Lin L. (1992) Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning.*

Lipton Z.C., Kale D.C., Eklan C., Wetzel R. (2017). Learning to Diagnose with LSTM Recurrent Neural Networks. *Machine Learning.*

Schaul T., Quan J., Antonoglou I., Silver D. (2016). Prioritized Experience Replay. *Machine learning.*

Simon H.A. (1990). Bounded Rationality. *Utility and Probability*

Tesfatsion L., Judd K. (2006). Handbook of Computational Economics. *Elsevier.*

Vamplew P., Dazely R., Foale C. (2017). Softmax exploration strategies for multiobjective reinforcement learning. *Neurocomputing.*

Watkins C., Dayan P. (1992). Technical Note: Q-Learning. *Machine Learning.*

Zhen S., Trott A., Srinivassa S., Naik N. et al. (2020). The AI Economist: Improving Equality and Productivity with AI-Driven Tax Policies. *General Economics.*