SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY —
DECISION SCIENCES & SYSTEMS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Engineering Science

Dynamics of Multi-Agent Reinforcement Learning Algorithms in a Discrete Time Oligopoly Price Competition

Ole Petersen

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY —
DECISION SCIENCES & SYSTEMS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Engineering Science

Dynamics of Multi-Agent Reinforcement Learning Algorithms in a Discrete Time Oligopoly Price Competition

Dynamik von Multi-Agent Reinforcement Learning-Algorithmen in einem zeitlich diskreten, oligopolistischen Preiswettbwewerb

Author:	Ole Petersen
Supervisor:	Prof. Dr. Martin Bichler
Advisor:	Fabian Pieroth
Submission Date:	12.10.2023

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 12.10.2023 Ole Petersen

Acknowledgments

First and foremost, I would like to thank my advisor Fabian Pieroth for his guidance and support throughout the entire thesis. It was only through his help that I was able to dive into the topic of multi-agent reinforcement learning and complete this thesis. His efforts went far beyond what I could have expected, taking the time to explain the concepts and ideas in detail and providing me with valuable feedback on my work.

I would also like to share my gratitude towards the chair of Decision Sciences & Systems, especially Prof. Dr. Martin Bichler, for providing me with the opportunity to write this thesis.

Finally, I would like to thank my family and friends for their support and encouragement through my studies, listening to my nerdy conversation in a (seemingly) interested manner.

Abstract

Multi-Agent Reinforcement Learning (MARL) has been proposed as a versatile tool for approximating Nash Equilibria (NE) in games where traditional techniques fail. In this study, we analyze the dynamic oligopoly model, a discrete-time game model for price competition in oligopolies, in multiple Reinforcement Learning (RL) experiments. We replicate the best response to the greedy strategy using Proximal Policy Optimization (PPO). Further, we apply Independent Reinforcement Learning (IRL) with PPO across diverse configurations. Our results highlight that a verified NE is attainable in both symmetric and small asymmetric setups, even when the dynamics are intricate. Surprisingly, in certain cases, identical agents adopt distinct strategies in the largest setup. This study suggests the promising capabilities of MARL in finding NE in complex games with continuous action spaces and multiple stages.

Contents

Notation

- For a set A, let $\Delta(A)$ denote the set of probability distributions over A.
- For a probability distribution $p \in \Delta(A)$, let $p(a)$ denote the probability of $a \in A$.
- For a probability distribution $p \in \Delta(A)$ and $B \subseteq A$, let $p(B)$ denote the probability of B under p, i.e. $p(B) = \sum_{a \in B} p(a)$.
- For a function $f : X \to \Delta(Y)$, $f(y|x)$ denotes the probability of $y \in Y$ given the input to f was $x \in X$.
- Given a quantity $x_{a,b,\ldots}$ that is indexed by a set of indices $a \in A, b \in B, \ldots$, let $x_{:,b,\ldots}$ denote the vector $(x_{a,b,\ldots})_{a \in A}$. Similarly, let $x_{a,:,\ldots}$ denote the vector $(x_{a,b,\ldots})_b$, and so on. If multiple indices are omitted, the result is a tensor.
- Given a quantity $x_{a,b,\ldots}$ that is indexed by a set of indices $a \in A, b \in B, \ldots$, let $x_{-i,b,\ldots}$ denote the vector $(x_{a,b,\ldots})_{a \in A \setminus i}$, similar to the definition above.
- δ_a denotes the Dirac delta function at a, i.e. $\delta_a(x) = \delta(x - a)$.
- For a random variable A, we write $\mathbb{E}_{a \sim A}[f(a)]$ for the expected value of $f(A)$, $\mathbb{V}_{a \sim A}[f(a)]$ for the variance of $f(A)$, and $\sigma_{a \sim A}(f(a))$ for the standard deviation of $f(A)$.

Nomenclature

Variable	Description
Reinforcement Learning	
S	State space
A	Action space
O	Observation space
$p : S \times A \to \Delta(S)$	Transition function
$r : S \times A \times S \to \mathbb{R}$	Reward function
$\sigma : S \to O$	Observation function
$\pi : O \to \Delta(A)$	Policy
$\rho_0 \in \Delta(S)$	Initial state distribution
$\gamma \in (0, 1]$	Discount factor
$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, s_2, \dots)$	Trajectory
$s_0(\tau)$, $a_0(\tau)$, $r_1(\tau)$, $s_t(\tau)$, $\dots$	Components of a trajectory
$G(\tau) = \sum_{t=0}^{\infty} \gamma^t r_{t+1}(\tau)$	Return of a trajectory
$V_\pi(s) = \mathbb{E}_{\tau \sim \pi \mid s_0 = s}[G(\tau)]$	Value of a state given a policy
$U(\pi) = \mathbb{E}_{s_0 \sim \rho_0}[V_\pi(s_0)]$	Utility of a policy
Dynamic Oligopoly	
n	Number of firms / agents
$\mathcal{N} = \{1, 2, ..., n\}$	Set of firms / agents
T	Number of rounds
$t \in \{1, 2, \dots, T\}$	Round index
$\tilde{t} \equiv T - t + 1 \in \{1, 2, \dots, T\}$	Number of rounds remaining
c_i	Production cost of firm i
C_i, $c_i \sim C_i$	Distribution of production cost of firm i
$p_t^i \in [c_i, D_t^i]$	Price of firm i in round t
$\Delta p_t^i \equiv p_t^i - \frac{1}{n} \sum_{j \in N} p_t^j$	Price difference of firm i at round t
D_t^i, $D_{t+1}^i \equiv D_t^i - \Delta p_t^i$	Demand of firm i in round t
$D_i \in \Delta(\mathbb{R}^+)$, $D_1^i \sim D_i$	Distribution of initial demand of firm i
$x_t^i \equiv D_t^i - p_t^i$	Sold quantity of firm i at round t
$r_t^i \equiv (p_t^i - c_i) x_t^i$	Profit of firm i at round t

$U_i \equiv \sum_{t=1}^{T} \rho^{t-1} r_t^i$	Discounted pay-off of firm i
$U_{i,t} \equiv \sum_{t'=t}^{T} \rho^{t'-1} r_{t'}^i$	Discounted pay-off of firm i from round t
$R_t^i \equiv D_t^i - c_i$	Relative demand potential of firm i at round t
$\lambda_{\tilde{t}}^i \in [0,1]$, $p_t^i = c_i + \lambda_{\tilde{t}}^i R_t^i$	Price indicator for agent i for $\tilde{t}$ rounds remaining
$\lambda_{\vdots}^{\vdots}$	Strategy profile
$U_i(\lambda_{\vdots}^{\vdots})$	Discounted pay-off of firm i given strategy profile $\lambda_{\vdots}^{\vdots}$

1. Introduction

Machine learning has seen tremendous progress in recent years, being praised as the future of our economy. While large language models finally reached the mainstream consumer market, the applications of machine learning go far beyond that, including the fields of computer vision, robotics, health care, and finance. In the latter, Reinforcement Learning (RL) has recently proven to be competitive against more traditional approaches by making use of the vast amounts of data available in the financial sector [1].

Game theory, on the other hand, is a far more established field of research that allows making rigorous statements about the interaction of multiple agents. Since the rigorousness of game theory is appealing in financial setups, it has been adopted widely in the field, with the most prominent example being auction theory [2], [3]. However, progress has been limited by the fact that often, none but the simplest financial models can be solved analytically, and even numerical methods are often infeasible.

Multi-Agent Reinforcement Learning (MARL) has been proposed as a versatile tool to overcome this limitation. Studies suggest that when multiple agents engage in simultaneous strategy learning, they often converge to a Nash Equilibrium (NE) [4].

We turn our attention to the dynamic oligopoly model as described by Bylka, Ambroszkiewicz, and Komar [5]. This discrete-time sequential game, which is based on price competition in an oligopoly, is sufficiently expressive to capture real-world phenomena. While simple versions of the model allow for clear analytical interpretation, more nuanced scenarios remain largely unexplored. Traditional methods find themselves limited, especially when faced with the game's continuous action and observation spaces, as well as its deep sequential characteristics. However, adapting this model into a MARL problem appears promising.

In our research, we undertake several RL experiments on the dynamic oligopoly environment to answer the following research questions:

- With which proximity can we reaffirm analytical results from both the original paper and our own work?
- Which novel insights can we gain from our experiments with verification?
- How do larger setups behave qualitatively?

We begin by giving an overview of work related to equilibrium computation. Then, we introduce the basics of game theory, RL, and equilibrium learning through IRL. The

dynamic oligopoly model, including the most significant analytical results, is presented next. Moving on, we describe the methodology of our experiments, followed by the results and an analysis thereof. We conclude with a summary of our findings and an outlook on future work.

2. Related Work

The quest for predicting game outcomes often pivots around the concept of a NE [6, Chapter 2.2]. Consequently, determining NE of various games has attracted significant research attention, yielding varied outcomes.

For discrete action space, normal-form games, identifying NE is computationally hard [7, Theorem 4.2.1]. However, when restricted to two-player zero-sum games, NE can be determined in polynomial time using linear programming [7, Chapter 4.1]. In perfect information extensive-form games, backward induction allows for NE identification in linear time w.r.t. the game tree size, as discussed in [7, Chapter 5.1.4]. Additionally, Bayesian normal-form games can be transformed into their perfect-information counterparts, thus inheriting the same solution methods [6, Definition 26.1]. Nonetheless, general-sum Bayesian extensive-form games demand exponential runtime in the worst case [7, Chapter 5.2.3].

Auction theory, a subset of game theory, offers profound insights into Bayesian games with infinite state and action spaces. While analytical solutions for first- and second-price sealed-bid auctions are well-understood territories [3], more intricate scenarios proved to be elusive. For instance, asymmetric first-price auctions boil down to a set of numerically challenging partial differential equations [8].

The fusion of machine learning and game theory has facilitated novel methodologies for identifying Nash Equilibria (NE). Initially devised for solving Markov Decision Processes (MDPs), RL was later explored for NE approximation by letting multiple agents simultaneously optimize their reward *independently*, therefore called Independent Reinforcement Learning (IRL) [9]. In a particular setting studied, many games demonstrated strategy profile convergence to a NE while some showed cyclic or even chaotic behaviors. Notably, distinct criteria under which Q-Learning would converge to a NE in this scenario were derived [4]. In recent work, Bichler, Fichtl, Heidekrüger, *et al.* [10] presented Neural Pseudogradient Ascent, a learning algorithm designed for identifying Bayes Nash Equilibria (BNE) in single-stage auctions with continuous observation and action domains.

RL has achieved major advances in various applications, including multi-stage situations with continuous actions and observations [11], [12]. This prompts the research community to question its applicability in multi-agent contexts.

3. Preliminaries

3.1. Game Theory

Game theory uses mathematical models to understand how rational players make strategic decisions. In this study, we use game theory concepts to describe and analyze how agents behave in a Multi-Agent Reinforcement Learning environment. In this section, we highlight the key game theory ideas that are relevant to our work. For a comprehensive grasp of game theory, Osborne and Rubinstein [6] serves as a valuable resource.

3.1.1. Normal-Form Games

Game theory examines different types of games, and a prominent category among them is the *normal form* or *strategic* game. In such games, each player selects an action from their available choices, unaware of the choices made by others. Once every player has made their decision, the game concludes, and players are rewarded based on the collective actions taken.

Osborne and Rubinstein [6, chapter 2.1] define a normal form game as follows:

Definition 1 (Normal-Form Game) *A* strategic game *consists of*

- *a finite set $\mathcal{N}$ (the set of players),*
- *for each player $i \in \mathcal{N}$, a nonempty set A_i (the set of actions available to player i), where $A = \times_{i \in \mathcal{N}} A_i$ is the set of action profiles or strategy profiles,*
- *for each player $i \in \mathcal{N}$, a utility function $U_i : A \to \mathbb{R}$, which the player tries to maximize.*

Two-player normal-form games can often be visualized using a payoff matrix. Take the game "Chicken" as described in [6, Example 16.3] for example. Imagine two players, Alice and Bob, driving head-on towards each other. They each face a choice: keep driving straight or swerve to avoid a collision. If one swerves while the other drives straight, the one who swerves (dubbed the "chicken") loses, earning a utility of -1. The straight driver triumphs with a utility of 1. Should both decide to swerve, both receive a utility of 0 — no collision occurs, but neither emerges as dominant. If both stay their course, disaster strikes, resulting in a collision and a utility of -10 for both.

This scenario is captured in Table 3.1, with each cell's first entry showing Alice's utility and the second showcasing Bob's.

	Bob	
Alice	Straight	Swerve
Straight	-10, -10	1, -1
Swerve	-1, 1	0, 0

Table 3.1.: Payoff matrix for the game "Chicken".

3.1.2. Mixed Strategies

The foundational idea discussed earlier presumes that players pick only one action. Yet, in many scenarios, it's more appropriate to think that players select from a range of actions based on certain probabilities. We term such a probabilistic approach a *mixed strategy*. Building on the standard normal-form game $(\mathcal{N}, A_{\cdot}, U_{\cdot})$, we can incorporate mixed strategies as described below [6, Chapter 3.1.1]:

- Player i selects a strategy through a probability distribution, denoted as π_i. This belongs to $\Delta(A_i) \equiv \Pi_i$, which represents the set of *strategies* or probability distributions for the potential actions for player i, A_i.
- The players' utility function U_i is extended by the expected value of the utilities of the pure outcomes i.e. $U_i(\pi_{\cdot}) = \mathbb{E}_{a_{\cdot} \sim \pi_{\cdot}}[U_i(a_{\cdot})]$.

3.1.3. Best Responses

Let's focus on player i and look at the strategies chosen by other participants, symbolized by π_{-i}. When we talk about a *best response*, we're referring to a strategy, π_i, that yields the highest utility for player i against the other players' strategies. Mathematically, this is given by [6, Chapter 2.2]:

$$U_i(\pi_i, \pi_{-i}) \geq U_i(\pi_i', \pi_{-i}) \quad \forall \pi_i' \in \Pi_i \tag{3.1}$$

3.1.4. Nash Equilibria

One primary concern in game theory is anticipating the strategies that players might adopt. Among several methods, the concept of a *Nash Equilibrium (NE)* stands out. At a NE, no player can get a better outcome by deviating from their strategy. More formally,

a Nash Equilibrium (NE) occurs when every player's strategy is the best response to the other players' strategies [6, Chapter 2.2]:

$$U_i(\pi_i, \pi_{-i}) \geq U_i(\pi_i', \pi_{-i}) \quad \forall i \in \mathcal{N}, \forall \pi_i' \in \Pi_i \tag{3.2}$$

Using the earlier "Chicken" game as a reference, two distinct pure NE arise: One where player A swerves and player B drives straight and another where player A drives straight while player B swerves. Notably, despite the game's symmetric structure, its equilibria are asymmetric.

3.1.5. ϵ-Nash Equilibria

In certain scenarios, finding a strategy profile where players can't notably benefit from slight deviations might suffice. Specifically, Shoham and Leyton-Brown [7, chapter 3.4.7] detail an ϵ-Nash equilibrium as:

Definition 2 (ϵ-Nash equilibrium) *Given an $\epsilon > 0$, a strategy profile $\pi_{:}$ qualifies as an ϵ-Nash equilibrium if*

$$U_i(\pi_i, \pi_{-i}) \geq U_i(\pi_i', \pi_{-i}) - \epsilon \quad \forall i \in \mathcal{N}, \forall \pi_i' \in \Pi_i$$

3.2. Reinforcement Learning

Game theory provides the foundation for the problem at hand, but our approach to solving it is rooted in Reinforcement Learning (RL). This section sheds light on the optimization challenge RL is meant to address, specifically the Markov Decision Process (MDP), and delves into the category of RL algorithms we use that facilitate this solution.

3.2.1. Markov Decision Processes

At its core, RL serves as an optimization mechanism for problems described by a Markov Decision Process (MDP). MDPs involve an *agent* navigating through an *environment* in a series of steps, each time aiming to optimize its *reward*.

To be more precise, consider an agent that, at every moment t, chooses an action $a_t \in A$ based on its observation $o_t \in O$ of the current state $s_t \in S$. The environment then draws from a probability distribution $p : S \times A \to \Delta(S)$ the next state $s_{t+1} \sim p(s_t, a_t)$, returning a reward $r_t = r(s_t, a_t, s_{t+1})$ to the agent [11, Chapter 3.1].

The agent's decision-making is modeled by a *policy* $\pi : O \to \Delta(A)$. If presented with an observation $o = \sigma(s)$ of a state s, the policy π will choose an action a with probability $\pi(a|o)$.

Iterating through this sequence of selecting actions and transitioning to ensuing states produces a *trajectory* defined as $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \dots)$, an ordered compilation of states, actions, and respective rewards. This dynamic is visualized in Algorithm 1.

An agent's overarching mission is to maximize the cumulative discounted rewards, defined as the return G_t:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.3}$$

We may also denote the return of a trajectory τ as $G(\tau)$. Using p and π, we can compute the probability of a trajectory τ given an initial state s. We write this as $\tau \sim \pi \mid s_t = s$, with the dependency of the trajectory on p being omitted.

Algorithm 1 Definition of the partially observable MDP. If the observation function σ is the identity function, the MDP is fully observable.

Require:
State space S
Action space A
Observation space O
Transition function $p : S \times A \to \Delta(S)$
Reward function $r : S \times A \times S \to \mathbb{R}$
Observation function $\sigma : S \to O$
Policy $\pi : O \to \Delta(A)$
Initial state distribution $\rho_0 \in \Delta(S)$
Discount factor $\gamma \in (0, 1]$

Initialize $s_0 \sim \rho_0$
Initialize trajectory $\tau = (s_0)$
for $t = 0, 1, 2, \dots$ **do**
    The agent observes $o_t = \sigma(s_t)$
    The agent chooses an action $a_t \sim \pi(o_t)$
    The state transitions to $s_{t+1} \sim p(s_t, a_t)$
    The agent receives reward $r_{t+1} = r(s_t, a_t, s_{t+1})$
    $\tau = \tau \cup (a_t, r_{t+1}, s_{t+1})$
end for
Reward the agent with $G = \sum_t \gamma^t \cdot r_t$

In the presented definition, the process continues indefinitely, and the discount factor γ ensures reward remains bounded [11, Equation 3.10]. However, in practice, it is

common to focus on *episodic* MDPs, where the iteration concludes after a set count, T, of steps. For consistency in notation, we describe an episodic MDP as follows, referencing Sutton and Barto [11, chapter 3.4]:

Definition 3 (Episodic MDP) *An episodic MDP with T steps is a MDP wherein after T steps, the state transitions to a designated terminal state, s_T, and all subsequent rewards post this transition are zero.*

3.2.2. The Reinforcement Learning Problem

The *value* of a state s given a policy π is defined as [11, Equation 3.12]:

$$V_\pi(s) = \mathbb{E}_{\tau\sim\pi|s_0=s}[G(\tau)] \tag{3.4}$$

Similarly, we can define the *action-value function* $Q_\pi(s,a)$ as the expected return when starting in state s, taking action a and then following policy π [11, Equation 3.13]:

$$Q_\pi(s,a) = \mathbb{E}_{\tau\sim\pi|s_0=s,a_0=a}[G(\tau)] \tag{3.5}$$

In RL, we aim to find a policy π^* that maximizes the expected return for all states $s \in S$:

$$\pi^* = \operatorname*{argmax}_\pi V_\pi(s) \tag{3.6}$$

In this formulation, we have a criterion for optimality for every possible initial state, which may be inconvenient. The initial state is sampled from a probability distribution $\rho_0 \in \Delta(S)$, allowing us to define an overall *performance coefficient* or *utility* $U(\pi)$ as [11, Equation 13.4]:

$$U(\pi) = \mathbb{E}_{s_0\sim\rho_0}[V_\pi(s_0)]. \tag{3.7}$$

The RL problem then becomes:

$$\pi^* = \operatorname*{argmax}_\pi U(\pi) \tag{3.8}$$

3.2.3. Policy gradient methods

There is a variety of RL algorithms, each with their strengths and weaknesses, see Sutton and Barto [11]. For the dynamic oligopoly environment, we require continuous action- and observation spaces, ruling out Deep Q Networks. Therefore, we choose Proximal Policy Optimization (PPO) [12] as our central RL procedure, which is a well-established actor-critic algorithm.

Actor-Critic Methods

These methods simultaneously try to approximate the optimal policy π^* and the policy value function V_π through interaction of the policy with the environment, making them model-free on-policy algorithms. Both the policy and the value function are approximated by a neural network, parameterizing the policy π_θ and the value function V_ϕ respectively. While only the policy is needed for the agent's decision-making, the value function helps in the training process. On a high level, the agent repeatedly interacts with the environment using its current policy π_θ and collects trajectories τ of states, actions, and rewards. These trajectories are then used to update the policy to make above-average-performing actions with a high *advantage* $\hat{A}_t$, as estimated using the value function, more likely [11, Chapter 13.5]. Finally, the value function is updated to better approximate the true value function V_{π_θ}. This procedure is repeated until convergence. Achiam [13, Vanilla Policy Gradient] describe the most basic actor-critic algorithm, the Vanilla Policy Gradient algorithm as in Algorithm 2.

Advantage Estimation

The advantage of an action a_t, $\hat{A}_t$, is an estimate for how good the performance of a_t is compared to the average performance of the policy π_θ. Ideally, we would like $\hat{A}$ to be the difference between the value of the action a_t and the value of the average action, i.e. $\hat{A}_t = q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t)$. However, since we do not know the true value function, we have to estimate the advantage. Here we have to make a trade-off between the stability of the algorithm and the bias introduced by inaccurate value functions [14, Section 2]. Schulman, Moritz, Levine, *et al.* [14] introduce Generalized Advantage Estimation (GAE), which allows to configure the bias-variance trade-off by a single parameter $\lambda \in [0, 1]$. The GAE advantage estimate $\hat{A}_t^{\mathrm{GAE}(\lambda)}$ is defined as follows:

$$\hat{A}_t^{\mathrm{GAE}(\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{3.9}$$

with $\delta_t = r_t + \gamma V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)$ being the temporal difference error. The parameter λ controls the bias-variance trade-off. For $\lambda = 1$, the advantage estimate is unbiased but has high variance, while for $\lambda = 0$, the advantage estimate has low variance but is biased.

Proximal Policy Optimization

Schulman, Wolski, Dhariwal, *et al.* [12] propose Proximal Policy Optimization (PPO), which is a refinement of the Vanilla Policy Gradient algorithm. It uses a clipped

Algorithm 2 Pseudocode for the Vanilla Policy Gradient algorithm by Achiam [13, Vanilla Policy Gradient]

Require:
Initial policy parameters θ_0
Initial value function parameters ϕ_0
MDP environment

for $k = 0, 1, 2, ...$ **do**
Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
Compute rewards-to-go $\hat{R}_t$.
Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.
Fit value function by regression on mean-squared error:

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

typically via some gradient descent algorithm.
end for

surrogate objective to prevent the policy from changing too much between updates. PPO is observed to significantly increase stability and performance compared to the Vanilla Policy Gradient algorithm [12, Conclusion].

Partial observability

The majority of RL algorithms operate under the presumption that the agent has complete visibility into the state s rather than relying on a partial observation, denoted as $o = \sigma(s)$. Therefore, partial observability breaks most convergence guarantees and should be kept in mind as a potential source of problems.

As a solution, techniques such as merging past actions and observations into an enhanced "super observation" or enabling the agent to accumulate an internal state via recurrent neural networks are now in play [15].

3.3. Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) takes the principles of single-agent RL a step further, enabling multiple agents to simultaneously engage with the environment. In this section, we'll delve into the foundational concepts of the MARL framework.

3.3.1. Markov games

Markov games, also known as *Stochastic Games*, adapt MDPs to accommodate multiple agents. Contrary to the setup presented by [7, Chapter 6.2], we define these games with the capability to encompass infinite state-, action-, and observation spaces, as illustrated in Algorithm 3.

Again, we define the *value* of a state given a strategy profile $\pi_:$ for player i as

$$V_{i,\pi_:}(s) = \mathbb{E}_{\tau \sim \pi_: | s_0 = s}[G_i(\tau)] \tag{3.10}$$

and the reward or *utility* of agent i as

$$U_i(\pi_:) = \mathbb{E}_{s_0 \sim \rho_0}[V_{i,\pi_:}(s_0)]. \tag{3.11}$$

We define an *episodic Markov game* in analogy to Definition 3.

Note that the agents' policies are independent of each other, given the observations, which is called a *behavior strategy* [7, Chapter 5.2.2].

It's often reasonable to assume that all agents retain a memory of the game's history—encompassing all its actions and observations. This assumption is termed *perfect recall* [7, Definition 5.2.3]. Although standard Markov games don't inherently possess

Algorithm 3 Definition of a Markov game. The game is partially observable if the observation function σ_i is not the identity function.

Require:
Set of agents $\mathcal{N}$
State space S
Action space $A = \times_{i \in \mathcal{N}} A_i$
Observation space $O = \times_{i \in \mathcal{N}} O_i$
Transition function $p : S \times A \to \Delta(S)$
Reward function $r : S \times A \times S \to \times_{i \in \mathcal{N}} \mathbb{R}$
Observation function $\sigma_i : S \to O_i$ for each agent $i \in \mathcal{N}$
Policy $\pi_i : O_i \to \Delta(A_i)$ for each agent $i \in \mathcal{N}$, where Π_i is the set of all policies for agent i
Initial state distribution $\rho_0 \in \Delta(S)$
Discount factor $\gamma \in (0, 1]$

Initialize $s_0 \sim \rho_0$
Initialize trajectory $\tau = (s_0)$
for $t = 0, 1, 2, \ldots$ **do**
  Each agent observes $o_t^i = \sigma_i(s_t)$
  Each agent chooses an action $a_t^i \sim \pi_i(o_t^i)$
  The state transitions to $s_{t+1} \sim p(s_t, a_t^i)$
  Each agent receives reward $r_{t+1}^i = r(s_t, a_t^i, s_{t+1})$
  $\tau = \tau \cup (a_t^i, r_{t+1}^i, s_{t+1})$
end for
Reward agent i with $G_i = \sum_t \gamma^t \cdot r_t^i$

perfect recall, given that agents are limited to observing the current state, we can conceptualize a version of Markov games that embodies perfect recall:

Definition 4 (Markov game with Perfect Recall) *Building upon the foundation of a Markov game, a* Markov game with perfect recall *retains the core structure of the original game but modifies the observation space. Specifically, the space O is supplanted by an extended observation space, represented as $\hat{O}_t = O \times \times_{\tau=0}^{t-1} A \times \times_{\tau=0}^{t-1} O$ at the time instance t. Consequently, the observation function σ_i is adapted to account for the accumulated history of observations and actions:*

$$\hat{o}_t^i = \hat{\sigma}_i(\tau) = (\sigma_i(s_t), a_0^i, a_1^i, \ldots, a_{t-1}^i, \sigma_i(s_0), \sigma_i(s_1), \ldots, \sigma_i(s_{t-1})). \tag{3.12}$$

This conceptual framework can be practically realized via methods delineated in Section 3.2.3, such as facilitating agents with the capability to maintain a recurrent

state.

In games with perfect recall, there is no loss in generality by restricting the policies to behavior strategies [7, Theorem 5.2.4].

Note that a Markov game where the strategies of all agents except one are fixed is a MDP for the remaining agent, which we call the *MDP corresponding to the Markov game with* π_{-i}.

3.3.2. The Multi Agent Reinforcement Learning Problem

In MARL, defining the objective of the learning process becomes more intricate compared to the straightforward goal in single-agent RL. One approach might be to focus on maximizing the combined returns of all agents, termed *social welfare*. This essentially reframes the multi-agent setup into a single-agent problem: a central entity assumes the role of all agents, optimizing their collective rewards by playing simultaneously [16, Chapter 5.3.1].

However, it is not always a sensible assumption that all agents are controlled by an altruistic dictator. For a more realistic view of the world, we are looking for a set of policies where each agent maximizes their individual return, given the other agents' policies, in other words, a NE:

Definition 5 (Nash Equilibrium of a Markov game) *A strategy profile* $\pi_{:}^{*}$ *is a* Nash equilibrium *of a Markov game if for all agents* $i \in \mathcal{N}$ *and all strategies* $\pi_i \in \Pi_i$

$$U_i(\pi_i^*, \pi_{-i}^*) \geq U_i(\pi_i, \pi_{-i}^*). \tag{3.13}$$

One category of algorithms, Independent Reinforcement Learning (IRL), achieves this equilibrium by having each agent adopt a single-agent learning mechanism *independently*, effectively treating other agents as facets of their environment [16, Chapter 5.3.2].

The foundational premise is simple yet profound: if every agent can flawlessly optimize their objectives (refer to Equation (3.8)), then the resulting strategy profile must be a NE.

Given that the agents will in all likelihood not be able to perfectly optimize their objectives, several performance metrics for IRL algorithms have been proposed:

- Convergence of expected return: How close does the expected return of each agent come to that of the Nash equilibrium? [16, Equation 5.4]
- Convergence of empirical action distribution: How close does the policy learned by each agent come to the Nash policy in terms of observation-action-pairs? [16, Equation 5.5]

- Convergence to an ϵ-NE: What is the maximum utility ϵ an agent loses compared to its best response?

3.3.3. Challenges in Multi-Agent Reinforcement Learning

It is observed that for some games, IRL simply does not converge to a NE. Several challenges have been identified:

- **Non-stationarity of the environment** In IRL, each agent assumes that the other agents' policies are fixed, but in reality, they are also learning. This means that the environment is constantly changing, which breaks the assumptions for the convergence of single agent RL algorithms, sometimes resulting in cyclic dynamics [17, Chapter 3.2].
- **Equilibrium Selection** Games often have multiple NE [6, Chapter 2.3], posing the additional challenge of coordinating to an equilibrium.
- **Scaling to Many Agents** An increase in the number of agents may not only increase resource requirements significantly but also aggravate the non-stationarity of the environment [17, Chapter 3.3].

3.4. Brute-force Verifying Best Responses

Given a Markov game and a strategy profile $\pi_{:}$, we can estimate the value of the best response of an agent, $U_i^*(\pi_{-i}) = \max_{\pi_i'} U_i(\pi_i', \pi_{-i})$ to π_{-i} by discretizing the action- and observation spaces and building up the full game tree. The value of this estimate may differ from its actual value in both directions since the verifier and the RL algorithm do not optimize over the same set of strategies.

The value may be underestimated if the discretization is too coarse and the highest-paying discretized strategy differs too much from the actual best response. It may on the other hand also be overestimated since by building up the game tree, the verifier strategies have perfect recall while the RL agent does not.

4. Dynamic Oligopoly

4.1. Definition

Bylka, Ambroszkiewicz, and Komar [5] propose the market model that is to be investigated in this work. It is a discrete-time oligopolistic price competition model that encapsulates the interactions among n firms or *agents* over T time periods. Each firm aims to maximize its profit by strategically determining the price of its product at every time step. The model is characterized by the procedure described in Algorithm 4.

Algorithm 4 Process of the dynamic oligopoly game as in Bylka, Ambroszkiewicz, and Komar [5].

Require:
Number of agents n
Number of rounds T
Discount factor $\gamma \in (0, 1]$
Initial demands D_1^i for each agent $i \in \mathcal{N} = \{1, 2, \ldots, n\}$
Production costs per unit c_i for each agent $i \in \mathcal{N}$

for $t = 1, 2, \ldots, T$ **do**
$\tilde{t} = T + 1 - t$
Compute relative demand potentials as $R_t^i = D_t^i - c_i$
Each agent i chooses its price $p_t^i = \lambda_{\tilde{t}}^i \cdot R_t^i + c_i \in [c_i, D_t^i]$
For each agent i compute the quantity sold as $x_t^i = D_t^i - p_t^i$
For each agent i compute the profit as $r_t^i = (p_t^i - c_i)x_t^i = \lambda_{\tilde{t}}^i \cdot (1 - \lambda_{\tilde{t}}^i) \cdot (R_t^i)^2$
Compute the average price as $\bar{p}_t = \frac{1}{n} \sum_{j \in N} p_t^j$
For each agent i compute the price difference $\Delta p_t^i = p_t^i - \bar{p}_t$
The demands transition to $D_{t+1}^i = D_t^i - \Delta p_t^i$
Each agent i drops out of the market if $D_{t+1}^i \leq c_i$
The remaining agents' demands are normalized, keeping total demand constant
end for
Reward each agent i with $U_i = \sum_{t=1}^{T} \gamma^{t-1} r_t^i$

The key assumptions are as follows:

- **Firms:** The agents represent firms that sell a uniform product. Customers make purchase decisions based purely on price.
- **Production Cost:** Each firm has a consistent production cost, denoted by c_i, per unit.
- **Demand Potential:** Each firm has a demand potential, D_t^i, representing the maximum number of customers inclined to purchase from that firm at time t.
- **Sales and Price:** A firm's sales volume x is inversely proportional to its price, given by $x_t^i = D_t^i - p_t^i$. For round t, the profit for firm i is $r_t^i = (p_t^i - c_i) \cdot x_t^i$.
- **Customer Migration:** Customers gravitate towards firms with competitive prices, reducing the demand potential for higher-priced firms in subsequent rounds. This is reflected in the equation $D_{t+1}^i = D_t^i - \Delta p_t^i$, where Δp_t^i signifies the price difference between firm i and the average price across all firms during round t.
- **Firm Removal:** Firms that can't operate profitably are eliminated. Their market presence is proportionally divided among the remaining firms. This occurs when $D_t^i \leq c_i$, as implied by $x_t^i = D_t^i - p_t^i \geq 0$ and $p_t^i \geq c_i$.

The relative demand potential for firm i in round t is $R_t^i = D_t^i - c_i$. This value signifies the size of the potential price range a firm might establish. A parameter called the *price inidicator* $\lambda_{\tilde{t}}^i \in [0, 1]$ is introduced, where $\lambda_{\tilde{t}}^i = 0$ signifies the minimal price c_i, and $\lambda_{\tilde{t}}^i = 1$ represents the maximum price D_t^i. The pricing rule for round t for firm i is thus $p_t^i = c_i + \lambda_{\tilde{t}}^i R_t^i$, where $\tilde{t} = T + 1 - t$ is the number of rounds remaining.

Firms must balance short-term and long-term gains. A short-term boost in profits by raising prices may result in a long-term profit decline due to diminishing market share.

4.2. Phrasing the model as a Markov game

To apply RL to the dynamic oligopoly, we formulate it as a Markov game.

Definition 6 (Markov game formulation of the dynamic oligopoly) *A tuple*

$$(n, T, \gamma, D_:, C_:, \alpha_:, \Omega_:)$$

is a dynamic oligopoly with n agents, T rounds, discount factor γ, and for each agent $i \in \mathcal{N}$ an initial demand distribution $D_i \in \Delta(\mathbb{R}^+)$, a production costs distribution of $C_i \in \Delta(\mathbb{R}^+)$, action parameterization $\alpha_i \in \{\lambda, p\}$, and observations $\Omega_i \subset \{\tilde{t} = 1, \tilde{t}, D_t^i, D_i, c_i, c_:\}$. It corresponds to the episodic Markov game $(\mathcal{N}, S, A_:, O_:, p, r_:, \sigma_:, \rho_0, \gamma)$ with the following variables:

- $\mathcal{N} = \{1, 2, \ldots, n\}$ *is the set of agents.*

- T *is the number of rounds of an episode.*
- $S = \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$ *is the state space. Each state* $s \in S$ *is a tuple* $(c_:, D^:, \tilde{t})$*, where* $c_:$ *is the vector of costs,* $D^:$ *is the vector of demands, and* $\tilde{t} = T + 1 - t$ *is the number of rounds remaining. We write* $c_:(s)$*,* $D^:(s)$*, and* $\tilde{t}(s)$ *to refer to the respective components of* s*.*
- $A_i = \begin{cases} [0,1] & \text{if } \alpha_i = \lambda \\ [c_i, D_{max}] & \text{if } \alpha_i = p \end{cases}$ *is the action space for agent* i*, with the maximum possible demand* D_{max}*. For* $a_i \in A_i$*, we write* $p_t^i(a_i)$ *to the price corresponding to* a_i*.*
- $O_i = \times_{obs \in \Omega_i} O^{obs}$ *is the observation space for agent* i*, where* O^{obs} *is the observation subspace for the observation obs.*
- $p : S \times A \to S$ *is the transition function. Since the transition is deterministic, it is a delta function so that for* $s' \sim p(s,a)$*, it holds that:*
  - $\tilde{t}(s') = \tilde{t}(s) - 1$
  - $c_:(s') = c_:(s)$
  - $D^i(s') = D^i(s) - \Delta p^i(s)$*, where* $\Delta p^i(s) = p^i(a_i) - \frac{1}{n}\sum_{j \in N} p^j(a_j)$*. If this resulted in* $D^i(s') \leq c_i$*, then* $D^i(s') = 0$ *and the demands of all remaining agents are normalized so that* $\sum_{j \in N} D^j(s') = \sum_{j \in N} D^j(s)$*.*
- $r_i(s,a,s') = (p_i(a) - c_i(s)) \cdot (D^i(s) - p_i(a)) = r_t^i$ *is the reward function for agent* i*.*
- $o_i = \sigma_i(s) \in O_i$ *is the observation for agent* i *of a state* s*. It is a subset of the state as indicated by*Ω_i*.*
- $\rho_0 \in \Delta(S)$ *is the initial state distribution so that if* $s_0 \sim \rho_0$*, it holds that* $c_i(s_0) \sim C_i$*,* $D^i(s_0) \sim D_i$ *for all* $i \in \mathcal{N}$*, and* $\tilde{t}(s_0) = T$*.*
- $\gamma \in (0,1]$ *is the discount factor.*

Given a trajectory $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, s_2, \dots)$*, we write* $p_t^i(\tau)$ *for* $p^i(a_t)$ *and* $D_t^i(\tau)$ *for* $D^i(s_t)$*.*

4.3. Analytical results

Bylka, Ambroszkiewicz, and Komar [5] find various analytical properties of their model, including the game dynamics for certain strategies, best responses for certain setups, and a NE in a restricted class of strategies.

4.3.1. Classification of Strategies

Agent i's profit in round t can be expressed as

$$r_t = \lambda_{\tilde{t}}^i (1 - \lambda_{\tilde{t}}^i)(R_t^i)^2 \tag{4.1}$$

reaching its maximum value for $\lambda = \frac{1}{2}$ [5, Equation 6]. It is found that a strategy for an arbitrary agent i can only be rational if $\lambda_t^i \in [0, 0.5] \forall t$ and $\lambda_1^i = \frac{1}{2}$ since playing $1 - \lambda_t^i$ would dominate playing λ_t^i if $\lambda_t^i > \frac{1}{2}$ [5, Lemma 3.1]. Therefore, from here on, only such strategies are considered. Additional terminology is introduced in [5, Definition 3.2] with a strategy $\lambda_\cdot^i$ being called

- greedy if $\lambda_{2:T}^i = 0$, maximizing demand
- myopic if $\lambda_\cdot^i = \frac{1}{2}$, maximizing short-term profit
- a stationary indicator strategy if $\lambda_{\tilde{t}}^i = l \ \forall \tilde{t} \geq 2$ for some $l \geq 0$.

4.3.2. Best responses

A recursive formula for an agent's best response to the greedy strategy is derived [5, Theorem 4.5]. For the special case of all agents having the same costs c, it is simplified to

$$\lambda_t^i = \begin{cases} \frac{1-\gamma+\gamma(2-e)\lambda_{t-1}^i}{2-\gamma e+\gamma e(2-e)\lambda_{t-1}^i} & \text{for } 1 < t \leq T \\ \frac{1}{2} & \text{for } t = 1 \end{cases} \tag{4.2}$$

with $e \equiv \frac{n-1}{n}$ [5, Theorem 4.9]. It further holds that

$$U_i = \alpha_T \cdot (R_1^i)^2 = \alpha_T (D_1^i - c_i)^2 \tag{4.3}$$

[5, Theorem 4.5] with

$$\alpha_t = \frac{1 - 2\lambda_{t+1}}{2e(1 - 2\lambda_{t+1})} \tag{4.4}$$

[5, Theorem 4.10]. For $\gamma = 1$ the expression simplifies further to

$$\lambda_t^i = \frac{1}{2 + (t-1)e} \tag{4.5}$$

resulting in a total profit of [5, Theorem 4.10]

$$U_i = \frac{T}{2(2 + Te - e)} (R_1^i)^2 \tag{4.6}$$

4.3.3. Symmetric Nash Equilibrium

We contribute the unique pure symmetric NE without demand observation and with absolute price action parameterization, which is given by

$$p_t^i = \frac{1}{2} \left(D + c - \frac{1}{\gamma^{t-1}} h_{t+1} \frac{n-1}{n} \right) \quad \forall i \in \mathcal{N} \tag{4.7}$$

with

$$h_t = \begin{cases} \gamma^{t-1}(p_t^i - c) + h_{t+1} & \text{for } 1 \leq t \leq T \\ 0 & \text{for } t = T+1 \end{cases} \tag{4.8}$$

where $c = c_i$ and $D = D_i$ are the costs and initial demand of the agents, respectively. This result is derived in Appendix A.1.

4.4. Symmetric Nash Equilibrium with demand observation

Similarly to the case above, we augmented the analytic results by the pricing outcome of a symmetric NE in the class of pure sub-game perfect strategies with demand observation. We show that the pricing strategy of any such NE can be approximated around the equilibrium point as

$$p_t^i(D_t^i) = p_t + k_t \cdot (D_t^i - D)$$

where

$$p_t = \frac{1}{2} \cdot (D + c - \frac{1}{\gamma^{t-1}} \cdot h_{t+1} \cdot \frac{n-1}{n})$$

$$h_t = \begin{cases} \gamma^{t-1} \cdot [(D - p_t) \cdot k_t + (1 - k_t) \cdot (p_t - c)] + h_{t+1} \cdot (1 - k_t) & \text{for } 1 \leq t \leq T \\ 0 & \text{for } t = T+1 \end{cases}$$

$$k_t = \frac{1}{2}$$

See Appendix A.2 for the derivation.

5. Methodology

We will carry out a series of experiments, all based on the foundational setup detailed in Section 5.1. Our metric collection process can be found in Section 5.2. For each experiment, we will outline its purpose, specify its parameters, and detail our expectations.

In every setup we discuss, parameters have been fine-tuned to yield optimal outcomes. For an in-depth look at this process, refer to Appendix A.3.

5.1. Setup

Our experiments are represented as Markov games, as elaborated in Section 4.2. This representation simplifies their implementation using PyTorch [18]. We employ IRL in a batched manner on the GPU, as showcased in Algorithm 5.

Algorithm 5 Procedure for a training run

```
Initialize the environment
Initialize the agents/learning algorithms
for each iteration do
    Sample a batch of new states at t = 1
    for each time step do
        Obtain agents' observations from states
        Derive agents' actions based on these observations
        Execute the actions
        Retrieve agents' rewards
        Transition to the next state
    end for
    Update the agents' learning algorithms
    Optionally, assess agents' policies and log results
end for
```

A *training run* refers to a single execution of Algorithm 5. An *experiment*, on the other hand, might compile results from numerous training runs. As an example, an experiment could repeat a training run with different values of a configuration

parameter to evaluate its influence on the results. We execute each configuration ten times using distinct random seeds to gauge the stability of our results, thus enabling us to determine *training uncertainties*.

Agents can be individually tailored with a learning algorithm, ranging from fixed strategies to various RL algorithms from stable baselines [19]. For our RL agents, we employ PPO [12], with the network architecture detailed in Appendix A.3.7. Note that we do not employ techniques such as a recurrent state to account for partial observability, potentially causing discrepancies with the verifier (see Sections 3.2.3 and 3.4).

5.2. Analysis

5.2.1. Collecting metrics

To collect metrics, trajectories are sampled from trained agents. These agents compete in the game without any policy updates, ensuring the training remains unattained from the metric estimation process.

Any given metric m can be approximated using a set of trajectories $\mathcal{T}$ derived from the agents' policies:

$$\mathbb{E}_{\tau\sim\pi_:}[m(\tau)] \approx \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}} m(\tau) \tag{5.1}$$

Empirical studies show that $|\mathcal{T}| = 2000$ provides a reliable estimation of the metrics in focus.

The learning outcome, which is the strategy profile $\pi_:$, inherently behaves as a random variable due to the stochastic characteristics of RL algorithms. We postulate that $\pi_:$ follows a distribution $P \in \Delta(\Pi_:)$. The *training uncertainty* associated with a metric m is defined as:

$$\sigma_{\pi_:\sim P}(\mathbb{E}_{\tau\sim\pi_:}[m(\tau)]) \tag{5.2}$$

This uncertainty can be approximated by repeatedly training with varied random seeds to sample different strategy profiles $\pi_: \sim P$. After estimating the metric m for each outcome using Equation (5.1) we can compute the standard deviation of the metric over the learning outcomes.

5.2.2. List of metrics

- **Price Indicator Mean** $\mathbb{E}_{\tau\sim\pi_:}[\lambda_t^i(\tau)]$: Represents agent i's average price indicator in round t. For a trajectory τ, it is denoted as $\lambda_t^i(\tau) = \frac{p_t^i(\tau)-c_i(\tau)}{D_t^i(\tau)-c_i(\tau)}$. $\lambda = 0$ implies

the lowest price, zero profit, and maximum future demand. Conversely, $\lambda = 1/2$ signifies the highest rational price, peak present profit, and maximum future demand loss. $\lambda = 1$ is reported for agents that drop out of the market for technical reasons.

- **Price Indicator Variance** $\mathbb{V}_{\tau\sim\pi_{:}}[\lambda_t^i(\tau)]$: Conveys the price indicator's variability for agent i in round t. If an agent's strategy is purely round-dependent, this serves as the *policy variance*.
- **Utility** $U_i = \mathbb{E}_{\tau\sim\pi_{:}}[G(\tau)]$
- **Utility Loss to the Best Response**: $U_i^* - U_i = \mathbb{E}_{\tau\sim\pi_i*,\pi_{-i}}[G_i(\tau)] - \mathbb{E}_{\tau\sim\pi_{:}}[G_i(\tau)]$ Calculated as the gap between agent i's best response utility and its actual utility. This metric indicates proximity to the best possible reward, useful for assessing if a strategy profile is an ϵ-NE. The best response utility may be estimated using analytical results specific to the setup, or the brute force verifier in the general case.
- **Utility Fraction of the Best Response**: $\frac{U_i}{U_i^*} = \frac{\mathbb{E}_{\tau\sim\pi_{:}}[G_i(\tau)]}{\mathbb{E}_{\tau\sim\pi_i*,\pi_{-i}}[G_i(\tau)]}$ Similar to the previous metric but in relative instead of absolute terms.

5.2.3. Role-based metrics

As described above, the strategy profile resulting from a training run is not deterministic. In certain scenarios, an agent's policy doesn't even roughly demonstrate consistent behavior across distinct random seeds. Instead, agents tend to adopt specific *strategy roles*. Take the "Chicken" game as an illustration: two agents might individually gravitate towards either the "Swerve" or "Straight" strategy role. While the consistency of these roles persists across multiple runs, which agent assumes a particular role remains unpredictable.

Given this behavior, it becomes more meaningful to aggregate metrics based on *roles* rather than individual *agents*. As an example, the price indicator mean for a role r_i in round t can be denoted as $\mathbb{E}_{\tau\sim\pi_{:}}[\lambda_t^{r_i}(\tau)]$.

To implement this, the allocation of roles to agents must be determined for each training run. While the specific assignment method varies by scenario, the consistent utility of a strategy may act as a reliable indicator for the role assignments.

5.3. Best response learning

We begin with an experiment with strong theoretical foundations; the best response learning experiment. Bylka, Ambroszkiewicz, and Komar [5] analytically derived the

best response to the greedy strategy (see Section 4.3.2) for a setup with symmetrical costs.

We try to replicate this result in its most general form through RL, randomizing the initial demand and testing multiple discount factors. By fixing the strategies of all agents except agent 0 to $\lambda = 0$ we reduce the problem to a MDP and can use PPO to learn the best response. While not strictly necessary, we allow the agent to observe its demand since this was observed to significantly improve the learning performance, for more details see Appendix A.3.

We employ the reward fraction of the best response U_0 / U_0^* and the price indicator by round $\lambda_:^0$ as metrics for convergence to the best response in terms of utility and pricing strategy, respectively.

Details of our experimental setup can be found in Table 5.2.

Parameter	**Value**
Dynamic oligopoly	
n	3
T	$3, 10$
γ	$\in \{0.5, 0.7, 0.9, 1.0\}$
$D_:$	$\mathcal{U}(2,5), \mathcal{U}(2,5), \mathcal{U}(2,5)$
$C_:$	$\delta_1, \delta_1, \delta_1$
action parameterization $\alpha_:$	$\lambda, \lambda, \lambda$
observations Ω_i	$\{\tilde{t}, D_t^i\}$
Other parameters	
strategies	PPO, $\lambda = 0$, $\lambda = 0$
iterations	300
num_envs	2000
learning rate	linear from 0.001 to 0
λ (GAE)	0.95
metrics	$\frac{U_0}{U_0^*}$, $\mathbb{E}[\lambda_:^0]$, $\mathbb{V}[\lambda_:^0]$

Table 5.2.: Parameters of the best response learning experiment

Given that our setup manifests as a deterministic, fully observable MDP, we anticipate that PPO will seamlessly converge to the best response. While not being groundbreaking on its own, being able to learn a best response reliably is a necessary prerequisite for more complex experiments involving NE. Beyond this, we aim to validate the efficacy of our dynamic oligopoly model representation as a Markov game. Moreover, we seek insights into the potential implications of the strategy not being

deterministic but a normal distribution with a small variance, at minimum.

5.4. Symmetric Nash Equilibrium learning

Our next step is to reproduce another analytical result: the symmetric NE as in Sections 4.3.3 and 4.4.

We replicate the analytical setup and employ IRL with PPO, as shown in Figure 5.1.

We would like to verify convergence to a NE in terms of utility and pricing strategy through the utility loss and the price indicator, respectively. However, no analytical best response to strategies other than the equilibrium strategy is known, making the brute force verifier our only method of verification in terms of utility. Since the verifier has a computational limit of three stages we run the experiment both for $T = 3$ and $T = 10$ rounds. The latter is not verifiable but might reveal further insights into the level of convergence in terms of pricing strategy.

Parameter	**Value**
Dynamic oligopoly	
n	3
T	3 or 10
γ	1
$D_:$	$\delta_5, \delta_5, \delta_5$
$C_:$	$\delta_4, \delta_4, \delta_4$
action parameterization $\alpha_:$	p, p, p
observations Ω_i	$\{\tilde{t}, D_t^i$ (optional)$\}$
Other parameters	
strategies	PPO, PPO, PPO
iterations	200
num_envs	7000
learning rate	linear from 0.001 to 0
λ (GAE)	$\begin{cases} 0.999 & \text{no demand observation} \\ 0.95 & \text{demand observation} \end{cases}$
metrics	$\frac{U_i}{U_i^*}\ \forall i \in \mathcal{N}$, $U_i^* - U_i\ \forall i \in \mathcal{N}$, $\mathbb{E}[\lambda_:]$, $\mathbb{V}[\lambda_:]$

Figure 5.1.: Parameters of the symmetric NE learning experiment

Unlike in the best response learning experiment, allowing the agents to observe their demand does modify the game. This is because an agent's demand is no longer computable from its own strategy alone but requires knowledge of the strategies of all

other agents. We investigate if demand observation increases the pricing, as predicted by our analytical results.

We expect the outcome of this experiment to be symmetric w.r.t. the agents given their identical setup. However, it remains to be seen how closely the learned strategies resemble the equilibrium strategy, given the complex dynamics of MARL.

5.5. Asymmetric Nash Equilibrium learning: Verifiable

Building upon the symmetric NE learning experiment, we introduce an asymmetry: we vary the costs of agent 0 while keeping other parameters constant (refer to Table 5.5). We will make use of the verifier, restricting the game to three stages without demand observation.

Parameter	Value
Dynamic oligopoly	
n	3
T	3
γ	1
$D_:$	$\delta_5, \delta_5, \delta_5$
$C_:$	δ_{c_0} $(c_0 \in [3.0, 4.5]), \delta_4, \delta_4$
action parameterization $\alpha_:$	p, p, p
observations Ω_i	$\{\tilde{t}\}$
Other parameters	
strategies	PPO, PPO, PPO
iterations	200
num_envs	7000
learning rate	linear from 0.001 to 0
λ (GAE)	0.999
metrics	$U_:$, $U_i^* - U_i \; \forall i \in \mathcal{N}$, $U_i / U_i^* \; \forall i \in \mathcal{N}$

Table 5.5.: Parameters of the verifiable asymmetric NE learning experiment

In our analysis, we prioritize examining the implications of this asymmetry on the learning outcome. Of all the potential metrics, we specifically focus on the utilities, which are the most important indicator of success for the agents. Additionally, we aim to ascertain whether a NE is achieved using the brute force verifier. While it is anticipated that any reduction in costs will be advantageous for agent 0, the precise dynamics and the ripple effects on other agents, especially in light of dropouts, are

subjects of our inquiry.

5.6. Asymmetric Nash Equilibrium learning: Large

Expanding upon the previous experiment, we extend the number of stages to $T = 10$. This aims to capture richer dynamics but comes at the expense of verifiability. Additionally, we permit demand observation to enhance stability, as detailed in Table 5.7.

Our observations reveal that agents probabilistically adopt *strategy roles*. Consequently, we present utility results *by role* rather than *by agent*. Specifically, agent 0 consistently selects r_0. In contrast, agents 1 and 2 fluctuate between r_1 (yielding higher profits) and r_2 (less profitable). The role choices of agents 1 and 2 in each training run are discerned based on their comparative utilities.

Parameter	Value
Dynamic oligopoly	
n	3
T	10
γ	1
$D_:$	$\delta_5, \delta_5, \delta_5$
$C_:$	δ_{c_0} $(c_0 \in [3.0, 4.5]), \delta_4, \delta_4$
action parameterization $\alpha_:$	p, p, p
observations Ω_i	$\{\tilde{t}, D_t^i\}$
Other parameters	
strategies	PPO, PPO, PPO
iterations	200
num_envs	7000
learning rate	linear from 0.001 to 0
λ (GAE)	0.999
metrics	U_{r_i} for roles r_0, r_1, r_2

Table 5.7.: Parameters of the large asymmetric NE learning experiment

6. Results

6.1. Best Response Learning

PPO shows a robust convergence towards the best response in both reward and strategy, as illustrated in Figure 6.1 and Table 6.1. Despite notable policy variance, the mean still aligns closely with the analytical deviation.

An outlier to this observation occurs in the later stages with a diminutive discount factor. Here, the mean price indicator falls notably short of the best response. Intriguingly, as the discount factor is reduced, the reward approaches its optimal value more closely.

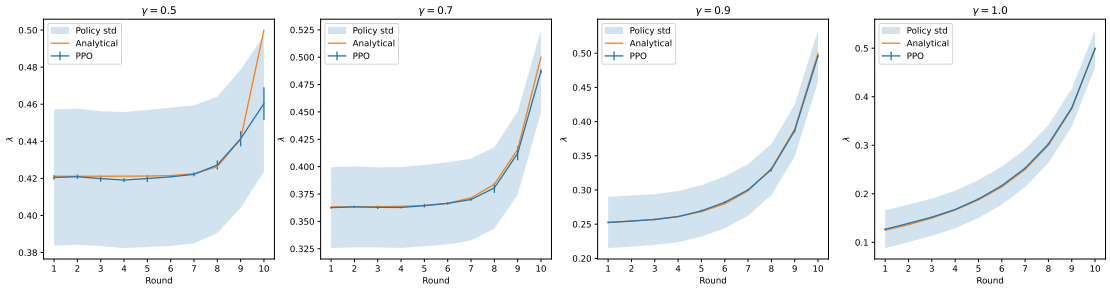


Figure 6.1.: Strategy learned by PPO for different discount factors. The training uncertainty is indicated by the error bars while the policy variance is illustrated by the shaded area.

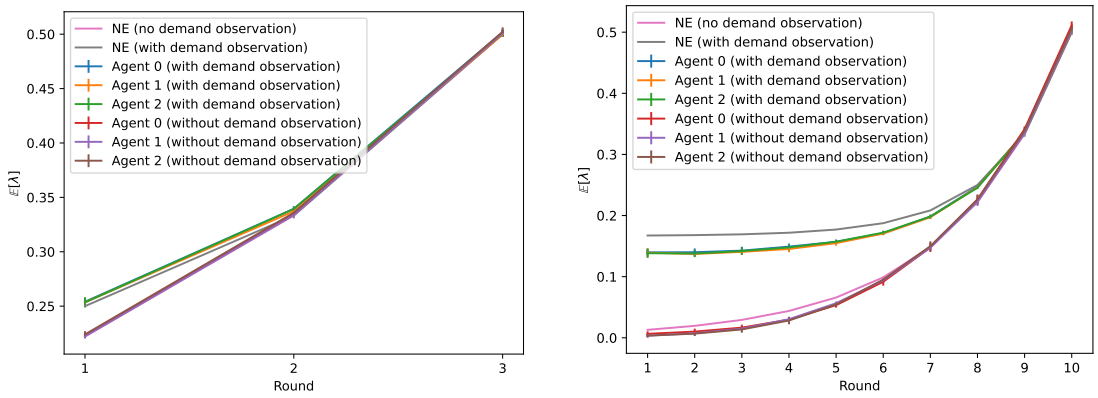
γ	U_0 / U_0^*
0.5	0.994805 ± 0.000389
0.7	0.994759 ± 0.000445
0.9	0.994079 ± 0.000451
1.0	0.991862 ± 0.000647

Table 6.1.: Utility achieved by PPO compared to its optimal value (computed analytically) for different discount factors.

6.2. Symmetric Nash Equilibrium Learning

When learning without demand observation, the derived policy aligns with the analytical model, as illustrated in Figure 6.2. For the 10-stage game, the incorporation of demand observation leads to elevated prices in initial rounds, matching our prediction. However, in the 3-stage scenario, this trend is only faintly evident. It is important to highlight that the fit, when observing demand, is not as precise compared to when demand is unobserved. This fit is strongly influenced by the GAE parameter λ (refer to Appendix A.3.3).

Remarkably, PPO identifies a policy that outperforms the verifier's estimate for the value of the best response by approximately 1%, as detailed in Table 6.2. This renders the strategy profile for $T = 3$ without demand observation as an approximate NE, accounting for discretization errors.



(a) $T = 3$ (b) $T = 10$

Figure 6.2.: Strategy learned by each agent in the symmetrical setup with and without demand observation, with the error bars indicating training uncertainties.

Metric	Agent 0	Agent 1	Agent 2
$U^* - U$	-0.006176 ± 0.004107	-0.006173 ± 0.004147	-0.006064 ± 0.004050
U/U^*	1.009694 ± 0.006474	1.009682 ± 0.006532	1.009526 ± 0.006397

Table 6.2.: Utility achieved by PPO for $T = 3$ without demand observation compared to its optimal value estimated by the verifier.

6.3. Asymmetric Nash Equilibrium learning: Verifiable

Figure 6.3 illustrates that as the costs for agent 0 rise, its profits correspondingly decrease. This relationship is near-perfectly represented by a quadratic fit, reaching its minimum at $c_0 = 4.62, U_0 = 0.02$. Notably, the only deviations from this fit occur around $c_0 \approx 4.35$.

Agents 1 and 2 display identical profits across all c_0 values, which is unsurprising given their identical setup. As the competitive advantage wanes, their profits exhibit a moderate increase. There's a pronounced jump from $U_1 = U_2 = 0.6 \mid c_0 = 4.3$ to $U_1 = U_2 = 3 \mid c_0 = 4.4$. It is worth noting that the uncertainties during the training phase are negligible for all cost values.

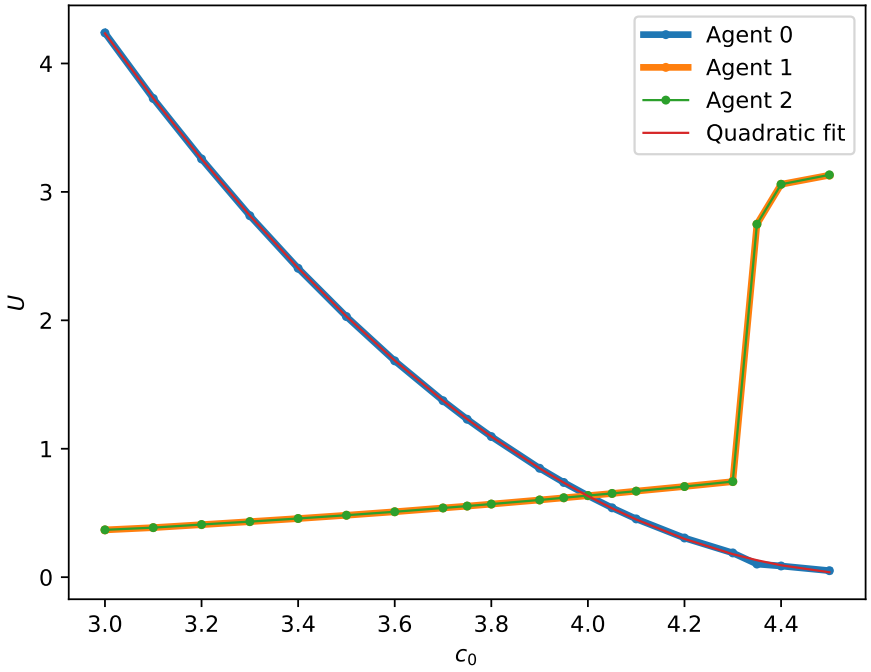
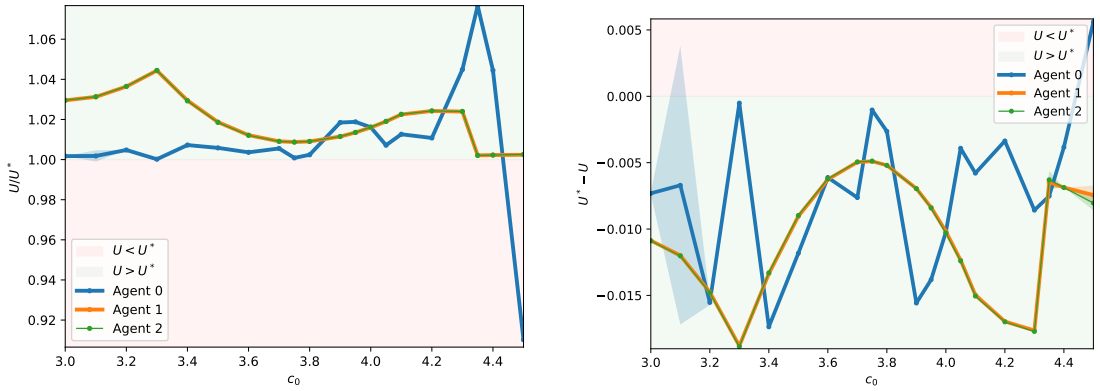


Figure 6.3.: Profits by agent over costs c_0 of agent 0 for $T = 3$. The shaded area indicates the training uncertainty, even though it is too small to be visible. The quadratic fit has the coefficients $U_0 \approx 1.61c_0^2 - 14.87c_0 + 34.36 = 1.61(c_0 - 4.62)^2 + 0.02$.

Further, Figure 6.4 demonstrates that, with the exception of $c_0 = 4.5$, PPO consistently outperforms the verifier's best response estimate. At $c_0 = 4.5$, the utility loss measures roughly 9% of the potential utility or 0.0056 in absolute terms. This positions the

strategy profile as a NE when $c_0 \neq 4.5$, and as a 0.0056-NE when $c_0 = 4.5$.



(a) Relative difference

(b) Absolute difference

Figure 6.4.: Utility achieved by PPO compared to its upper bound estimated by the verifier for different costs c_0 at $T = 3$. The shaded area indicates the training uncertainty. Discretization errors may lead to negative utility losses, refer to Section 3.4 for details.

6.4. Asymmetric Nash Equilibrium learning: Large

Figure 6.5 suggests that the game dynamics for the "large" version retain qualitative similarities to its "small" counterpart near the symmetrical point. As agent 0's competitive advantage grows, its profits rise, while the profits for the other agents drop, notably experiencing a jump when $c_0 = 4.1$.

However, in quantitative terms, the game showcases far more pronounced behavior. There is an increased sensitivity to cost changes, and overall profits as well as their range are higher.

Delving into the dynamics for large c_0 values, we observe that agent 0's profits plunge from $U_0 = 1 \mid c_0 = 4.05$ to $0.5 \mid c_0 = 4.1$. Conversely, profits for the other agents surge from $U_1 = U_2 = 1.5 \mid c_0 = 4.05$ to $U_1 = U_2 = 9 \mid c_0 = 4.1$. Beyond $c_0 = 4.1$, agents 1 and 2's profits stabilize at $U_1 = U_2 = 9$, while agent 0's profits steadily decline, dropping to $U_0 = 0.05$ by $c_0 = 4.5$.

Notably, agent 0's profit sees a sharp escalation from $3 \mid c_0 = 3.8$ to $80 \mid c_0 = 3.7$. Further reduction in costs to $c_0 = 3.0$ results in approximately a 2.5-fold decrease in its profit.

For agents 1 and 2 smaller c_0 values unveil a unique pattern, absent in the smaller

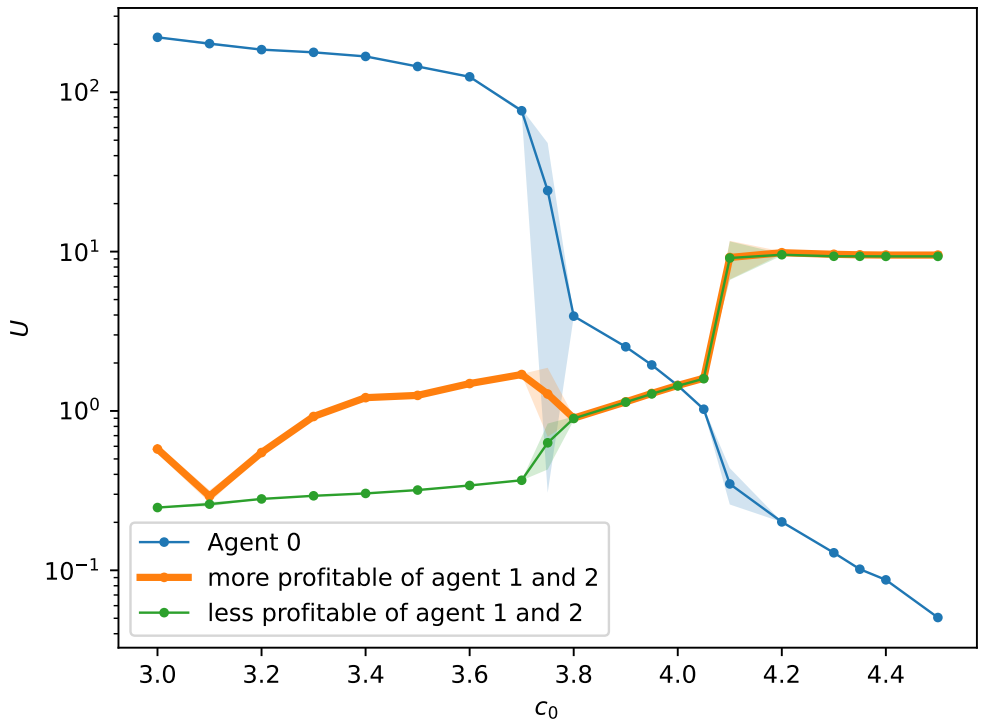


Figure 6.5.: Profits in relation to the costs c_0 of agent 0 for $T = 10$. The shaded area indicates training uncertainty.

version of the game. Within the interval $c_0 \in [3.3, 3.7]$, either agent 1 or agent 2 achieves a significantly higher profit than the other, despite their symmetry. Agents 1 and 2 learn the higher- and lower-profit strategy roughly equally often for different random seeds, with a distribution of 113 : 87 over all seeds and values of c_0. The more profitable agent achieves roughly the same profit in all training runs, but it varies between runs which agent this is, with similar results for the less profitable agent.

To be more precise, agents 1 and 2 probabilistically adopt one of two *strategy roles* during the training. The agent picking role 1 yields around 5 times the profit of that with role 2. By $c_0 = 3.1$, these roles converge, only to diverge again when c_0 reaches 3.0. Intriguingly, when c_0 is increased from 3.8 to 3.7, the agent adopting the more profitable role benefits from its more severe competitive disadvantage.

Training uncertainties remain negligible with exceptions only in the regions around $c_0 = 4.1$ and $c_0 = 3.8$.

7. Analysis

7.1. Best response learning

In summary, the training proved to be effective. The response learned aligns well with the analytical best response identified by Bylka, Ambroszkiewicz, and Komar [5], both in terms of reward and strategy. The alignment of the average price indicator with the best response is not inherently evident considering the model's non-linearity. The consistent policy variance relative to the round index can be attributed to the observation-independent standard deviation, as illustrated by the policy network architecture in Figure A.9.

We hypothesize that intense discounting might reduce the strength of the signals reaching the algorithm in later rounds, resulting in the deviation from the optimum strategy that was observed. To gauge the peak signal strength, we evaluated the maximum profit attainable in each round, predicated on the agent's strategy prior rounds. A diminished maximum profit implies the agent's actions exert minimal influence on the reward, leading to a weakened signal.

As depicted in Figure 7.1, both the maximal achievable profit and, correspondingly, the agent's signal, are diminished for minor discount factors during later stages. These zones coincide with significant deviations from the best response in Figure 6.1, substantiating that a reduced signal strength could rationalize the suboptimal performance of PPO in these domains.

To conclude, PPO adequately learns the best response, with minor exceptions for extremely low discount factors. Yet, even in these instances, the discrepancies are justifiable. This is a promising result for more complex setups, where analytical results are not as strong as in this case.

7.2. Symmetric Nash Equilibrium Learning

In the verifiable version of this setting ($T = 3$, without demand observation), we discerned that agents effectively learn a NE to a precision matching the verifier's discretization error.

Moreover, the strategies adopted by agents align closely with our analytical predic-

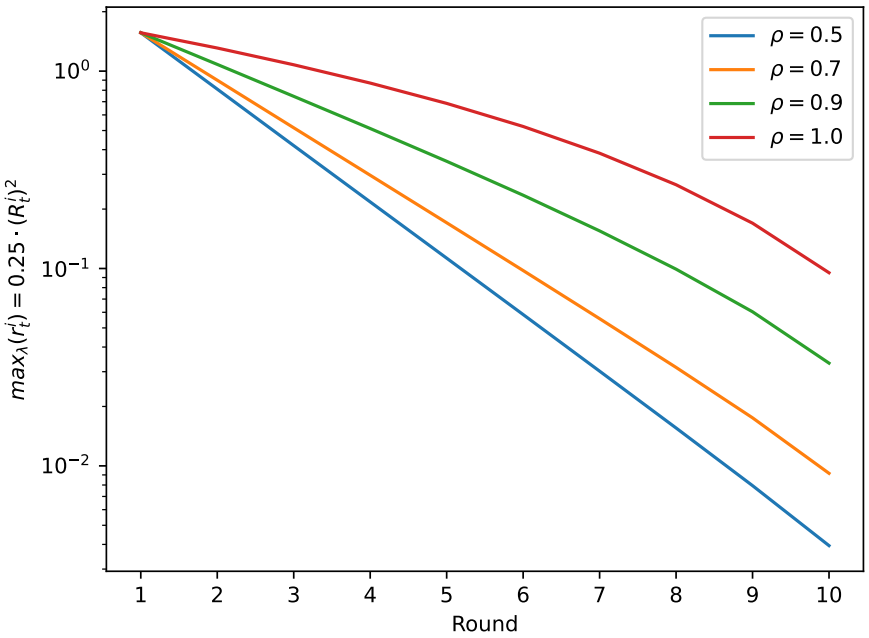


Figure 7.1.: Maximum profit achievable per round in the best response learning, assuming the agent adopts the strategy learned in preceding rounds. The round's peak profit results from deploying $\lambda = 1/2$, equating to $r_t^i = 0.5 \cdot (1 - 0.5) \cdot (R_t^i)^2 = 0.25 \cdot (R_t^i)^2$ (refer to Equation (4.1)).

tions for this context. The observation that agents adopt precisely this strategy, coupled with the brute force validation of their strategies as a NE, reinforces our conviction that a NE is indeed learned.

In the scenario with $T = 10$ void of demand observation, the strategies pursued by agents continue to align with our analytical expectations. Given the parallels with the verifiable context, it is conceivable that agents are navigating towards a NE here as well. Notwithstanding, the multiplicity of stages prevents direct verification.

Transitioning to experiments incorporating demand observation, we discern a pattern: the additional observation steers towards higher pricing. This difference magnifies from neutrality in the terminal round to its peak in the first round. The outcome matches our analytical prediction; to gain further insights into the rationale, we contrast two polarized strategy profiles: the "greedy" and the "myopic" strategy profiles.

The "greedy" profile requires no cooperation to work; no agent can suffer a loss from another agent deviating. However, it is not favorable to the agents since they make no profit in any but the last round.
Conversely, the "myopic" profile emerges as the antithesis: it yields the highest total profits in the class of symmetric strategy profiles but requires the agents to believe that the other agents will cooperate. Each agent can easily deviate, driving the other agents out of the market and dramatically increasing its own profit. This strategy profile would require an altruistic dictator (from the perspective of the companies, not the consumers) to work.

In the absence of insights into their current demand, agents are blindsided regarding competitor strategies in-play. Consequently, they learn a strategy that is close to the greedy strategy profile, with price indicators close to zero in the first rounds. With demand observation, on the other hand, the agents have a way of testing if the other agents are cooperating. This makes it less risky to play higher prices in early rounds since the agents can react to deviations of their competitors, avoiding being driven out of the market.

7.3. Asymmetric Nash Equilibrium learning: Verifiable

Firstly, the jump in the profits of agents 1 and 2 at $c_0 = 4.3$ is striking. This is suspected to originate from the discontinuity of agents opting out of the game. To validate this presumption, we examined the average demand per agent for each round immediately before and after the jump, as illustrated in Figure 7.2. Observably, while on the "left" side of this jump the demands subtly lean towards agents 1 and 2, they manage to edge the disadvantaged agent 0 out of the market on the "right" side. As agent 0 exits the scene, the demand potential it previously held becomes available for the

remaining contenders, significantly boosting their profits. Yet for agent 0, the dent in profit remains relatively minor as it was already teetering close to zero.

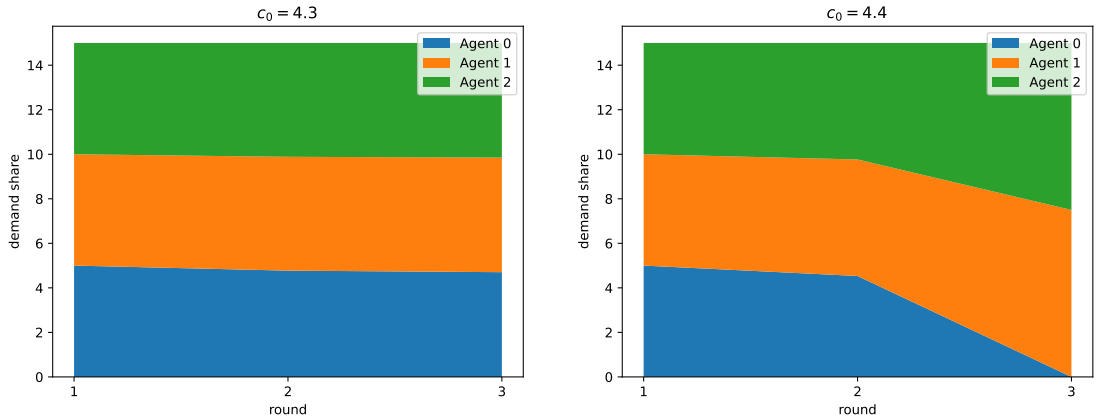


Figure 7.2.: Average demands per round for each agent with cost values flanking the jump at $c_0 = 4.3$ in the verifiable asymmetric NE learning experiment.

Further, the precision of the quadratic fit in tracing the correlation between agent 0's profits and its costs is intriguing, particularly around the jump. While a complete understanding remains elusive, some preliminary insights suggest a potential rationale for this observed relationship.

Bylka, Ambroszkiewicz, and Komar [5] identified numerous scenarios where an agent's profit displayed a quadratic dependence on its relative demand potential, evident in equations like Equation (4.1) and Equation (4.3). Such relationships often presuppose other agents following a predetermined strategy. While that might not precisely align with our current setup, it's an approximation that carries weight: Should agent 0 bear significantly lower costs than its counterparts, they would gravitate towards their minimum price, i.e. their invariable production costs. The congruence of the vertex at $c_0 = 4.62, U_0 = 0.02$ with a relationship directly proportional to the squared initial relative demand potential (i.e., $c_0 = 5.0, U_0 = 0$) fortifies this perspective.

One might initially conjecture that the jump at $c_0 = 4.3$ would compromise the quadratic fit. However, its diminutive magnitude prevents any drastic impact. Nonetheless, when visualized on a logarithmic scale, the error in the fit emerges as more pronounced.

In attempting to understand the significant relative utility loss for agent 0 at $c_0 = 4.5$ when using PPO, we put forth the following potential explanations:

- **Insufficient signal:** Despite the relative utility loss being pronounced, the absolute loss remains small. This could result in weak signals reaching the algorithm,

similar to the scenario detailed in Section 7.1.

- **No perfect recall:** Unlike the verifier, the agents do not have perfect recall, as described in Section 3.4. The lack of information might be especially determinant in this scenario.
- **Ineffective learning:** The learning process might simply have been ineffective in this particular instance.

7.4. Asymmetric Nash Equilibrium learning: Large

In this experiment, we observe several intricate phenomena. We focus on elucidating two: the profit jumps w.r.t. c_0, and the disparity between agents 1 and 2 despite their identical setup.

7.4.1. Explaining the jumps in the agents' profits

As laid out in Section 6.4, the profits of the agents exhibit two jumps at $c_0 = 3.75$ and $c_0 = 4.1$. We hypothesize that these jumps arise due to agents leaving the game, like in Section 7.3. To validate this assumption, we plotted the number of rounds agents stay in the game for different values of c_0 in Figure 7.3.

A key observation is the alignment between profit jumps and the instances when agents exit the game. For the interval $c_0 \in [3.75, 4.1]$, no agents are removed. Notably, the jumps occur precisely at this interval's endpoints.

We also posit that the training uncertainty spikes at these boundaries because, in some training seeds, agents may remain while in others, they might exit.

7.4.2. Explaining the asymmetry of agents 1 and 2

Despite their identical setup, agents 1 and 2 probabilistically adopt distinct *strategy roles* with diverging payoffs in the interval $c_0 \in [3.3, 3.7]$, as described in Section 6.4. To delve deeper, we visualized their price indicators in Figure 7.4. These roles can be categorized as:

- **Role 0 (Agent 0's role)**: This agent initially uses small lambda values, driving its competitors out of the game.
- **Role 1 ("Straight")**: Taken by either agent 1 or 2, this role involves playing small lambda values at the game's outset and converting its demand potential into profit midway, thereby leaving the game. The agent adopting this role reaps substantial profits if it can capitalize on the other agent dropping out of the game.

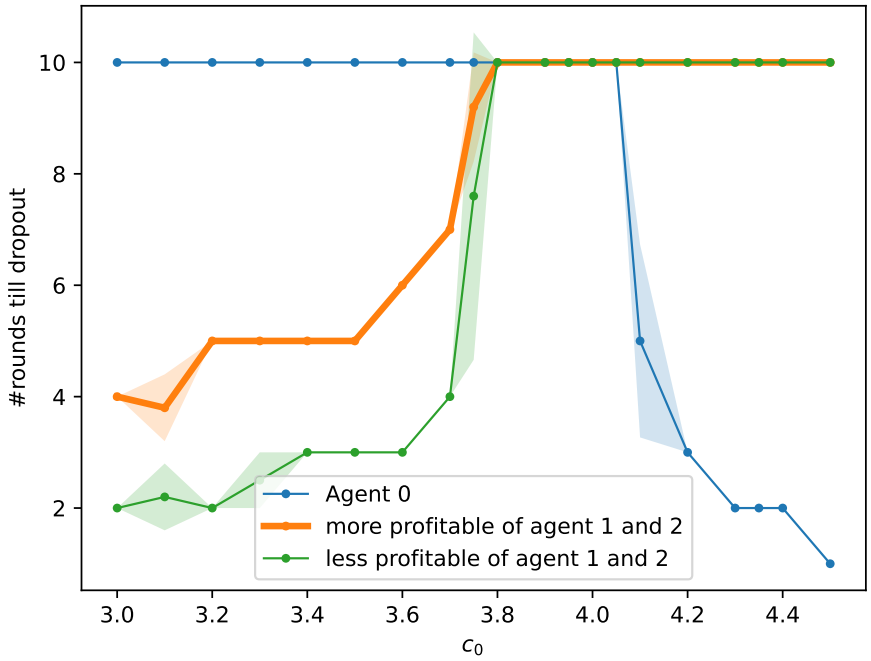


Figure 7.3.: Number of rounds agents stay in the game for varying values of c_0 in the large NE learning setup. The shaded area indicates the training uncertainty. The number of rounds an agent stays in the game is estimated by the number of rounds an agent has an average demand that is higher than its costs.

We term this the "straight" strategy since it relies on the other agent "swerving" and dropping out of the game.

- **Role 2 ("Swerve")**: The alternative role for agents 1 or 2. Here the agent deploys large lambda values right from the beginning, guaranteeing moderate profits at the cost of an early exit after the third round. We call this the "swerve" strategy due to its risk-averse nature.

It is plausible that agent 0 invariably profits from eliminating other agents, thereby amplifying its initial relative demand potential from $R = 5 - 4 = 1$ to $R = 3 \times 5 - 4 = 11$, especially since profit scales quadratically with the relative demand potential. Thus, we'll treat agent 0's strategy as a constant and shift our attention to agents 1 and 2.

To find a reason for agents 1 and 2 choosing different strategies, we formulated a simplified payoff matrix for these agents, displayed in Table 7.1. This matrix, built upon the strategy roles learned, approximates the agents' decision-making process when

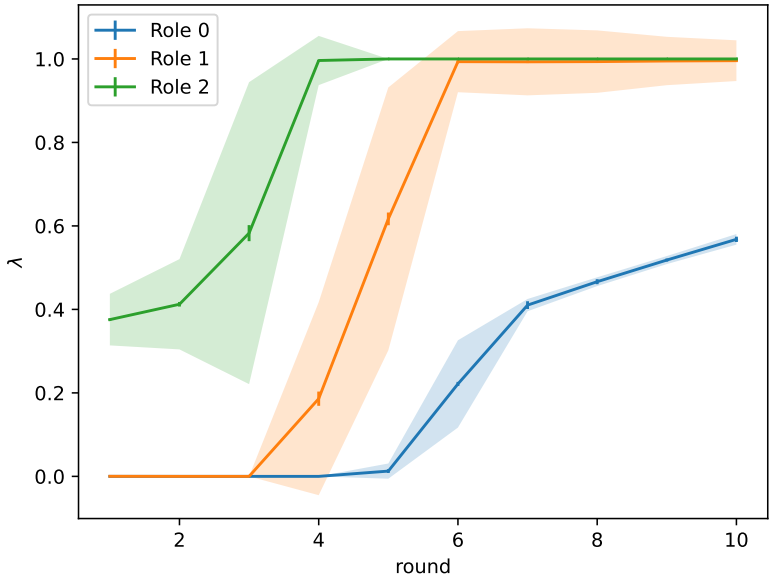


Figure 7.4.: Strategy roles learned as in Section 5.6 for $c_0 = 3.5$, with the shaded area illustrating $\mathbb{V}[\lambda_t^{r_i}]$ and the error bars indicating the training uncertainty. Role 0 is that of agent 0 while roles 1 and 2 are that of the more- and less profitable of agents 1 and 2, respectively. $\lambda = 1$ is reported if agents drop out of the game. Further, agents can observe the demand in this setup, therefore the shaded area is significantly higher than the policy variance.

faced with the choice of "straight" or "swerve".

	Agent 2	
Agent 1	Straight	Swerve
Straight	0.064, 0.064	0.583, 0.315
Swerve	0.315, 0.583	0.378, 0.378

Table 7.1.: Payoff matrix for the reduced game of the setup in Section 5.6 for $c_0 = 3.5$. It is simplistically assumed that agents 1 and 2 can only choose between the "straight" and "swerve" strategies and that agent 0 always plays strategy role 0. We approximate all three strategy roles as constant-lambda-by-round strategies with the lambda distributions being approximated as normal distributions with the means and variances of the price indicators learned by the agents. Note this does not fully match the strategies learned by PPO since the algorithm can observe the demand. This analysis methodology dramatically simplifies the agents' decision problem; in the real game, agents can both vary their pricings during the game based on demand observations and choose from a far larger set of strategies. Therefore, this reduction can only give a qualitative intuition of the nature of the game.

A key insight from our analysis is the resemblance of this game to the classic "Chicken" game described in Section 3.1.1. In this scenario, only one of agents 1 and 2 can effectively counter agent 0 over a substantial duration. The "straight" strategy relies on the other agent adopting the "swerve" approach, maintaining a high average price, enabling the "straight" player to persist. Conversely, both playing "straight" leads to a shared downfall since both players will be eliminated without ever capitalizing on their demand potential.

Drawing parallels with "Chicken", this offers an intuitive grasp of the strategies' asymmetry. The pure Nash equilibria in "Chicken", [straight, swerve] and [swerve, straight], align with our learning outcomes. While "Chicken" also presents a mixed Nash equilibrium, its feasibility remains in question given the policy's reliance on normal distributions. Despite our analysis being a simplification, it renders the observed outcomes conceivable.

8. Conclusion

In this work, we translated the market model introduced by Bylka, Ambroszkiewicz, and Komar [5] into a Markov game, allowing us to apply RL to it.

We learned the best response to the greedy strategy that was derived analytically with single-agent RL. In a symmetrical setup, we found a NE through IRL. This NE is verified using a brute-force verifier in a small case, and it matches our analytical considerations. We observed that the learning outcome depends on the observation space in that transparency increases the pricing.

Once we break the symmetry, we cannot rely on analytical results anymore, and the dynamics increase in complexity. In a small setup, we found a brute-force-verified NE through IRL again, for different levels of asymmetry. The expectation that a competitive advantage leads to a higher profit was confirmed, and we observed that an increase in the competitive advantage of one agent harms the other agents, even if their absolute setup remains unchanged. Once an agent is disadvantaged too much, it drops out of the game, causing a discontinuous increase in the remaining agents' profits.

When increasing the number of rounds, the results get even more complex, at the cost of verifiability. We observed a behavior similar to the smaller case, but with two "jumps" in the agents' profits w.r.t. their competitive advantage, all of which can be traced back to agents dropping out of the game. Most surprisingly, two identical disadvantaged agents were observed to stochastically adopt distinct strategy roles, one of which is significantly more profitable than the other. Even though we cannot fully explain this phenomenon, we give an intuition for the asymmetry using an analogy to the game "Chicken".

We conclude that IRL with PPO achieved its learning objective in all the cases we considered if configured properly. This makes us optimistic that MARL can also be employed to approximate NE in other scenarios where traditional approaches fail. However, more efficient verification methods are needed to make the results of increasingly complex setups with more than three stages meaningful.

In our study of the dynamic oligopoly model, its apparent simplicity belied the emergence of diverse phenomena, most of which would have been hard to discover using traditional methods. Although we deepened our understanding of the model with several new insights, we maintain that this is just an initial foray into the potential of MARL in this area.

Abbreviations

RL Reinforcement Learning

BNE Bayes Nash Equilibrium

MDP Markov Decision Process

NE Nash Equilibrium

MARL Multi-Agent Reinforcement Learning

PPO Proximal Policy Optimization

GAE Generalized Advantage Estimation

IRL Independent Reinforcement Learning

List of Figures

List of Tables

Bibliography

[1] B. Hambly, R. Xu, and H. Yang, "Recent advances in reinforcement learning in finance," en, *Mathematical Finance*, vol. 33, no. 3, pp. 437–503, 2023, ISSN: 1467-9965. DOI: 10.1111/mafi.12382.

[2] A. V. Thakor, "Game Theory in Finance," *Financial Management*, vol. 20, no. 1, pp. 71–94, 1991, Publisher: [Financial Management Association International, Wiley], ISSN: 0046-3892. DOI: 10.2307/3666098. [Online]. Available: https://www.jstor.org/stable/3666098 (visited on 09/18/2023).

[3] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," en, *The Journal of Finance*, Mar. 1961.

[4] M. Wunder, M. Littman, and M. Babes-Vroman, "Classes of Multiagent Q-learning Dynamics with ϵ-greedy Exploration," in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, Aug. 2010, pp. 1167–1174.

[5] S. Bylka, S. Ambroszkiewicz, and J. Komar, "Discrete time dynamic game model for price competition in an oligopoly," en, *Annals of Operations Research*, vol. 97, no. 1, pp. 69–89, Dec. 2000, ISSN: 1572-9338. DOI: 10.1023/A:1018900913350.

[6] M. J. Osborne and A. Rubinstein, *A course in game theory*, en. Cambridge, Mass: MIT Press, 1994, ISBN: 0-262-65040-1.

[7] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. USA: Cambridge University Press, 2008, ISBN: 0521899435.

[8] G. Fibich and N. Gavish, "Numerical simulations of asymmetric first-price auctions," en, *Games and Economic Behavior*, vol. 73, no. 2, pp. 479–495, Nov. 2011, ISSN: 08998256. DOI: 10.1016/j.geb.2011.02.010.

[9] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," *Morgan Kaufmann*, pp. 330–337, 1993. DOI: https://doi.org/10.1016/B978-1-55860-307-3.50049-6. [Online]. Available: https://web.media.mit.edu/~cynthiab/Readings/tan-MAS-reinfLearn.pdf.

[10] M. Bichler, M. Fichtl, S. Heidekrüger, N. Kohring, and P. Sutterer, "Learning equilibria in symmetric auction games using artificial neural networks," en, *Nature Machine Intelligence*, vol. 3, no. 8, pp. 687–695, Aug. 2021, ISSN: 2522-5839. DOI: 10.1038/s42256-021-00365-4.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction* (Adaptive computation and machine learning), en. Cambridge, Mass: MIT Press, 1998, ISBN: 978-0-262-19398-6.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv*, Aug. 2017, arXiv:1707.06347 [cs]. DOI: 10.48550/arXiv.1707.06347.

[13] J. Achiam, *Spinning Up in Deep Reinforcement Learning*, 2018. [Online]. Available: https://spinningup.openai.com (visited on 09/01/2023).

[14] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, *High-Dimensional Continuous Control Using Generalized Advantage Estimation*, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1506.02438.

[15] L. Meng, R. Gorbet, and D. Kulić, *Memory-based Deep Reinforcement Learning for POMDPs*, arXiv:2102.12344 [cs], Sep. 2021. DOI: 10.48550/arXiv.2102.12344. (visited on 09/05/2023).

[16] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2023. [Online]. Available: https://www.marl-book.com.

[17] K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*, Apr. 2021. [Online]. Available: http://arxiv.org/abs/1911.10635 (visited on 09/18/2023).

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, 2019. arXiv: 1912.01703.

[19] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html.

A. Appendix

A.1. Derivation of the Symmetric Nash Equilibrium

Consider a dynamic oligopoly in its Markov game formulation with symmetric production costs c and initial demand D where the agents can only observe the round index $\tilde{t}$, i.e.

Parameter	**Value**
n	n
T	T
γ	γ
$D_:$	$\delta_D, \delta_D, \ldots$
$C_:$	$\delta_c, \delta_c, \ldots$
action parameterization $\alpha_:$	$p, p, \ldots$
observations Ω_i	$\{\tilde{t}\}$

Pure strategy profiles can be parameterized by $p_:^:$ so that $\pi_i(o_i) = \delta_{p^i_{\tilde{t}(o_i)}}$. For symmetric NE, all agents have the same demand ($D_t^i = D_t \ \forall i$) in each round, and since total demand is constant, $D_t^i = D_t = D \ \forall i, t$. Since demand is constant, no agents drop out of the game and U_i is a differentiable function of $p_:^:$. Being a Nash equilibrium requires the pricing strategy $p_:^i$ to be optimal given the other agents' strategies, and therefore $\partial_{p_t^i} U_i = 0 \ \forall t$. Recall the definition of U_i and $U_{i,t}$:

$$U_i = \sum_{t=1}^{T} \gamma^{t-1} r_t^i$$
$$U_{i,t} = \sum_{t'=t}^{T} \gamma^{t'-1} r_{t'}^i$$

We can use this to derive a recursive formula for the symmetric NE:

$$\begin{aligned}
\partial_{p_t^i} U_i &= \\
\partial_{p_t^i} \sum_{t'=1}^{T} \gamma^{t'-1} r_{t'}^i &= \\
\frac{\partial_{p_t^i} \sum_{t'=1}^{t-1} \gamma^{t'-1} r_{t'}^i}{\partial p_t^i} + \frac{\partial r_t^i \gamma^{t-1}}{\partial p_t^i} + \frac{\partial U_{i,t+1}}{\partial p_t^i} &= \\
0 + \frac{\partial r_t^i \gamma^{t-1}}{\partial p_t^i} + \frac{\partial U_{i,t+1}}{\partial D_{t+1}^i} \frac{\partial D_{t+1}^i}{\partial p_t^i} &= \\
\gamma^{t-1} \frac{\partial (p_t^i - c_i)(D_t^i - p_t^i)}{\partial p_t^i} + \frac{\partial U_{i,t+1}}{\partial D_{t+1}^i} \frac{\partial D_t^i - (p_t^i - \frac{1}{n} \sum_{j \in N} p_t^j)}{\partial p_t^i} &= \\
\gamma^{t-1}(D_t^i - 2p_t^i + c_i) - \frac{\partial U_{i,t+1}}{\partial D_{t+1}^i} \frac{n-1}{n} &\stackrel{!}{=} 0
\end{aligned}$$

It can be seen that U_i is a quadratic function of p_t^i with a negative leading coefficient, and therefore $\partial_{p_t^i} U_i = 0$ is not just a necessary but also a sufficient condition for a global maximum.

Introducing the alias $h_t \equiv \partial_{D_t^i} U_{i,t}$, we can rewrite the above equation as

$$p_t^i = \frac{1}{2} \left(D_t^i + c_i - \frac{1}{\gamma^{t-1}} h_{t+1} \frac{n-1}{n} \right) \tag{A.1}$$

We set $h_{T+1} = 0$, using the fact that $U_{i,T+1} = 0$, being an empty sum. For $t \leq T$, we can recursively derive h_t as follows:

$$\begin{aligned}
h_t &= \partial_{D_t^i} U_{i,t} \\
&= \frac{\partial r_t^i \gamma^{t-1}}{\partial D_t^i} + \frac{\partial U_{i,t+1}}{\partial D_{t+1}^i} \frac{\partial D_{t+1}^i}{\partial D_t^i} \\
&= \gamma^{t-1} \frac{\partial (p_t^i - c_i)(D_t^i - p_t^i)}{\partial D_t^i} + h_{t+1} \frac{\partial D_t^i - (p_t^i - \frac{1}{n} \sum_{j \in N} p_t^j)}{\partial D_t^i} \\
&= \gamma^{t-1}(p_t^i - c_i) + h_{t+1} \cdot 1
\end{aligned}$$

Consequently, we have

$$h_t = \begin{cases} \gamma^{t-1}(p_t^i - c_i) + h_{t+1} & \text{for } 1 \leq t \leq T \\ 0 & \text{for } t = T+1 \end{cases} \tag{A.2}$$

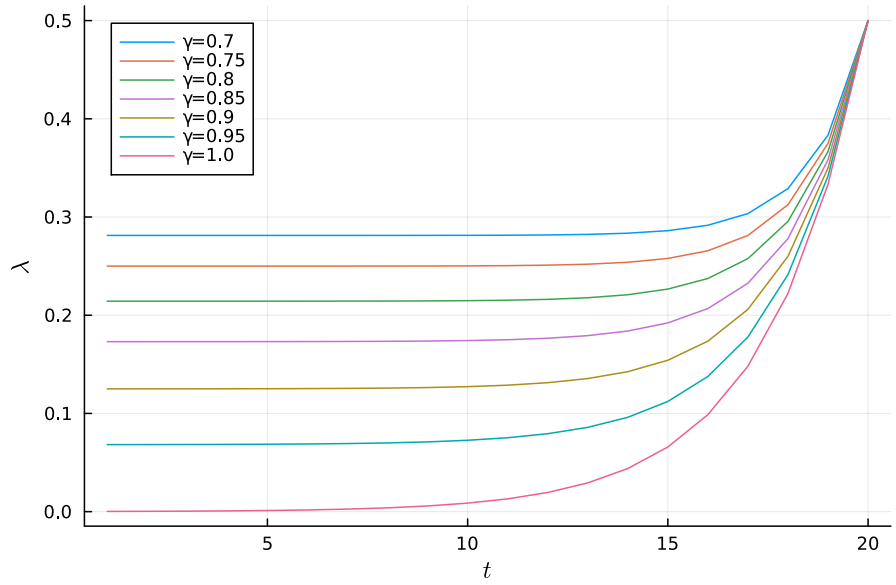


Figure A.1.: Symmetric NE without demand observation for $T = 20$, $n = 3$, $D = 5$, $c = 4$ as derived analytically for different values of γ

Using $\lambda_t^i = \frac{p_t^i - c_i}{D_t - c_i}$, the result of the recursive formula has been plotted in Figure A.1. It can be seen that the further the game progresses, the more the firms focus on short-term profits, i.e. increase their prices. As expected, a smaller discount factor γ causes firms to focus more on short-term profits, i.e. increase their prices more quickly.

For $\gamma = 1$, it can be seen that the price indicator approaches zero in all but the last rounds.

A.2. Derivation of the Symmetric Nash Equilibrium with demand observation

Building on the previous result, we will now derive the symmetric NE with demand observation:

Parameter	**Value**
n	n
T	T
γ	γ
$D_:$	$\delta_D, \delta_D, \ldots$

$C_:$	$\delta_c, \delta_c, \ldots$
action parameterization $\alpha_:$	$p, p, \ldots$
observations Ω_i	$\{\tilde{t}, D_t^i\}$

With demand observation, the sequential nature of the game comes into play. The strategy space explodes, consisting of all combinations of functions of the price given the demand observed for each round for each agent. With this explosion of strategies also comes an explosion of NE. For instance, the strategy profile where all agents play the myopic strategy unless they observe a deviation, in which case they play the greedy strategy, is a NE. Therefore, we will make the following assumptions:

- **Symmetry:** The pricing strategy is the same for all agents
- **Pure strategies:** The agents do not randomize their actions.
- **Continuity:** For each agent $i \in \mathcal{N}$, the pricing strategy p_t^i is a continuous function of the demand observed D_t^i in all rounds. Around the equilibrium point, the strategy can be approximated by a linear function

  $$p_t^i = p_t + k_t \cdot \Delta D_t^i$$

  where p_t is the price at the equilibrium point (the point where $D_t^i = D \ \forall i, t$) and $\Delta_t^i = D_t^i - D$ is the deviation of the observed demand from the equilibrium demand.
- **Sub-game perfection:** At each point t of the game, each agent plays a strategy that maximizes its future reward $U_{i,t}$.
- **Linearity of value:** The value of a state s_t for an agent i is fully described by its demand D_t^i, i.e. it does not matter for an agent how the remaining demand is distributed among the other agents.

We will start with some basic observations:

$$D_t^i = D + \Delta D_t^i \tag{A.3}$$
$$p_t^i = \bar{p}_t^i + k_t^i \cdot \Delta D_t^i \tag{A.4}$$
$$\bar{p}_t^i = p_t \ \forall i \in \mathcal{N} \tag{A.5}$$
$$k_t^i = k_t \ \forall i \in \mathcal{N} \tag{A.6}$$

and therefore if $t \leq T$:

$$U_{i,t} = \gamma^{t-1} \cdot (D_t^i - p_t^i) \cdot (p_t^i - c) = \gamma^{t-1} \cdot (D + \Delta D_t^i - p_t - k_t \cdot \Delta D_t^i) \cdot (p_t + k_t \cdot \Delta D_t^i - c) \tag{A.7}$$

Next, it holds that

$$\begin{aligned}
\frac{\partial D_{t+1}^i}{\partial D_t^i} &= \frac{\partial D_t^i + \sum_{j \in \mathcal{N}} (p_t^j / n) - p_t^i}{\partial D_t^i} \\
&= 1 + \frac{\partial \sum_{j \in \mathcal{N} \setminus \{i\}} (p_t^j / n)}{\partial D_t^i} + \partial_{D_t^i}(p_t^i / n) - k_t \\
&= 1 + \frac{1}{n} \frac{\partial \sum_{j \in \mathcal{N} \setminus \{i\}} (p_t + D_t^j \cdot k_t)}{\partial D_t^i} - k_t \cdot \frac{n-1}{n} \\
&= 1 + \frac{k_t}{n} \frac{\partial \sum_{j \in \mathcal{N} \setminus \{i\}} (D_t^j)}{\partial D_t^i} - k_t \cdot \frac{n-1}{n} \\
&= 1 + \frac{k_t}{n} \frac{\partial (D \cdot n - D_t^i)}{\partial D_t^i} - k_t \cdot \frac{n-1}{n} \\
&= 1 - \frac{k_t}{n} - k_t \cdot \frac{n-1}{n} \\
&= 1 - k_t
\end{aligned} \tag{A.8}$$

Note this is the major change compared to the previous section, where we had $\frac{\partial D_{t+1}^i}{\partial D_t^i} = 1$. Allowing agents to react to demand changes reduces their ability to "take demand from one round to the next".

Further, we have

$$\frac{\partial D_{t+1}^i}{\partial p_t^i} = \partial_{p_t^i}(D_t^i + \sum_{j \in \mathcal{N}} (p_t^j / n) - p_t^i) = -\frac{n-1}{n} \tag{A.9}$$

Now, we are ready to state our first condition of optimality: At the equilibrium point, no agent should have an incentive to deviate. Since in equilibrium, $\Delta D_t^i = 0 \; \forall i, t$, we have

$$\begin{aligned}
0 =& \partial_{\bar{p}_t^i} U_{i,t} \\
=& \partial_{\bar{p}_t^i}(\gamma^{t-1} \cdot (D - \bar{p}_t^i) \cdot (\bar{p}_t^i - c)) + \partial_{\bar{p}_t^i} U_{i,t+1} = \quad \gamma^{t-1} \cdot (D - 2\bar{p}_t^i + c) + \frac{\partial U_{i,t+1}}{\partial D_{t+1}^i} \cdot \frac{\partial D_{t+1}^i}{\partial \bar{p}_t^i} \\
=& \gamma^{t-1} \cdot (D - 2\bar{p}_t^i + c) - h_{t+1} \cdot \frac{n-1}{n}
\end{aligned}$$

with $h_t = \partial_{D_t^i} U_{i,t}$ as defined in the previous section. This yields

$$p_t = \bar{p}_t^i = \frac{1}{2} \cdot (D + c - \frac{1}{\gamma^{t-1}} \cdot h_{t+1} \cdot \frac{n-1}{n}) \tag{A.10}$$

Now, let us find an expression for h_t:

$$\begin{aligned}
h_t =& \partial_{D_t^i} U_{i,t} \\
=& \partial_{D_t^i} (\gamma^{t-1} \cdot (D + \Delta D_t^i - p_t - k_t \cdot \Delta D_t^i) \cdot (p_t + k_t \cdot \Delta D_t^i - c)) + \partial_{D_t^i} U_{i,t+1} \\
=& \gamma^{t-1} \cdot [(D + \Delta D_t^i - p_t - k_t \cdot \Delta D_t^i) \cdot k_t + (1 - k_t) \cdot (p_t + k_t \cdot \Delta D_t^i - c)] + \frac{\partial U_{i,t+1}}{\partial D_{t+1}^i} \cdot \frac{\partial D_{t+1}^i}{\partial D_t^i}
\end{aligned}$$

and therefore at the equilibrium point using Equation (A.8)

$$h_t = \gamma^{t-1} \cdot [(D - p_t) \cdot k_t + (1 - k_t) \cdot (p_t - c)] + h_{t+1} \cdot (1 - k_t) \tag{A.11}$$

Now we come to the final condition: sub-game optimality. If agents maximize their reward in each sub-game, they do so for small deviations, too. Optimal behavior in the sub-game is equivalent to choosing k_t^i optimal given the deviation ΔD_t^i:

$$\begin{aligned}
0 =& \partial_{k_t^i} U_{i,t} \\
=& \partial_{k_t^i} (\gamma^{t-1} \cdot (D + \Delta D_t^i - p_t - k_t \cdot \Delta D_t^i) \cdot (p_t + k_t \cdot \Delta D_t^i - c)) + \partial_{k_t^i} U_{i,t+1} \\
=& \gamma^{t-1} \cdot \Delta D_t \cdot (D + \Delta D_t^i - p_t - k_t \cdot \Delta D_t^i - p_t - k_t \cdot \Delta D_t^i + c) + \frac{\partial U_{i,t+1}}{\partial D_{t+1}^i} \frac{\partial D_{t+1}^i}{\partial p_t^i} \frac{\partial p_t^i}{\partial k_t^i} \\
=& \gamma^{t-1} \cdot \Delta D_t \cdot (\Delta D_t^i - 2k_t \cdot \Delta D_t^i + h_{t+1} \frac{n-1}{n} \frac{1}{\gamma^{t-1}}) - h_{t+1} \cdot \frac{n-1}{n} \cdot \Delta D_t^i
\end{aligned}$$

with the last equality holding since Equation (A.10) $\Leftrightarrow D - 2p_t + c = h_{t+1} \frac{n-1}{n} \frac{1}{\gamma^{t-1}}$. Therefore, dividing by $\gamma^{t-1} \cdot \Delta D_t$, we have

$$\Delta D_t^i - 2k_t \cdot \Delta D_t^i + h_{t+1} \frac{n-1}{n} \frac{1}{\gamma^{t-1}} = h_{t+1} \cdot \frac{n-1}{n} \frac{1}{\gamma^{t-1}}$$

and consequently

$$k_t = \frac{1}{2} \tag{A.12}$$

Putting it all together (Equations (A.10) to (A.12)), we have the recursive formula:

$$p_t = \frac{1}{2} \cdot (D + c - \frac{1}{\gamma^{t-1}} \cdot h_{t+1} \cdot \frac{n-1}{n}) \tag{A.13}$$

$$h_t = \begin{cases} \gamma^{t-1} \cdot [(D - p_t) \cdot k_t + (1 - k_t) \cdot (p_t - c)] + h_{t+1} \cdot (1 - k_t) & \text{for } 1 \leq t \leq T \\ 0 & \text{for } t = T + 1 \end{cases} \tag{A.14}$$

$$k_t = \frac{1}{2} \tag{A.15}$$

The result is plotted in Figure A.2. It can be seen that demand observation causes the price indicator to stabilize at a constant value for $\gamma = 1$ in the early rounds. This is in contrast to the case without demand observation, where price indicators approach zero in all but the final few rounds.

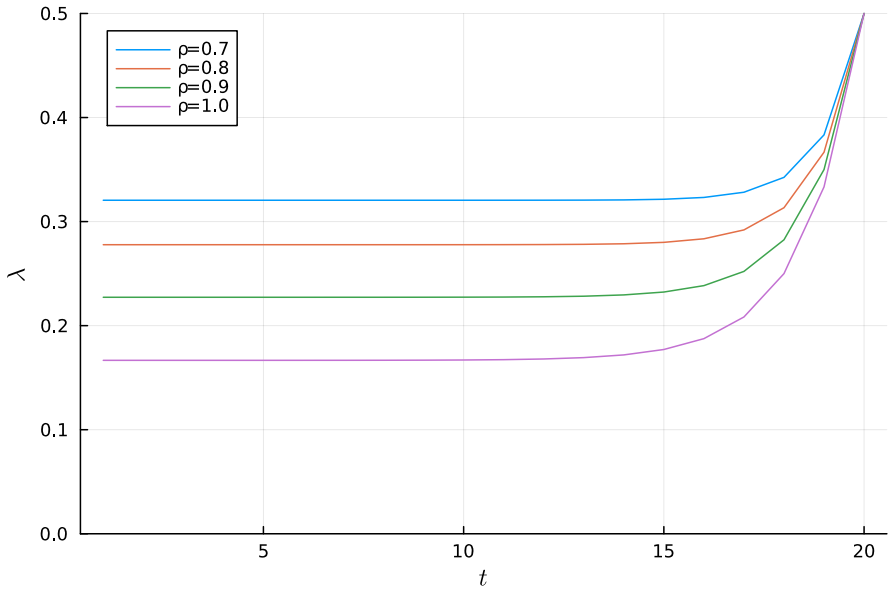


Figure A.2.: Symmetric NE for $T = 20$, $n = 3$, $D = 5$, $c = 4$ with demand observation as derived analytically for different values of γ

As a check of plausibility we compute the pricing strategy in the last round. Here, it is known that playing $\lambda = 1/2$ is optimal:

$$
\begin{aligned}
p_T^i &= p_T + k_T \cdot \Delta D_T^i \\
&= \frac{1}{2} \cdot (D + c - \frac{1}{\gamma^{T-1}} \cdot h_{T+1} \cdot \frac{n-1}{n}) + \frac{1}{2} \cdot (D_t^i - D) \\
&= \frac{1}{2} \cdot (D + c) + \frac{1}{2} \cdot (D_t^i - D) \\
&= \frac{1}{2} \cdot (c + D_t^i) \\
&= c + \underbrace{\frac{1}{2}}_{\lambda^*} \cdot (D_t^i - c)
\end{aligned}
$$

A.3. Fine-tuning Parameters

A.3.1. Learning rate and number of iterations

A good learning rate was found in the best response learning experiment. We varied the rate to find a value that allows for the fastest stable convergence. We then selected an iteration number so that training stops once the agent does not improve anymore. The utility by iteration in this setup for the base experiment is shown in Figure A.3.

Another measure we used to verify convergence is *numerical policy gradients*. When the means of the policy learned did not match the analytical best response for initial configurations we wanted to test if the variance of the policy was causing this deviation. A possible explanation would have been that, given the fact that the policy always has at least some standard deviation, the optimal mean of the policy *given its variance* would not match the optimal mean *for variance zero*. To test this, we calculated the numerical gradients of the utility w.r.t. the policy means, given the standard deviation. The results in Figure A.4 that for the properly configured setup, the gradients approach zero in the last iteration.

A.3.2. Demand observation and GAE

At some point, we found that the agent overshoots the best response price in initial rounds when the demand is not observed in the best response learning experiment. We discerned the reason for this is the bias introduced by the GAE [14]. Recall the definition of the GAE in Section 3.2.3:

$$
\hat{A}_t^{\mathrm{GAE}(\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{A.16}
$$

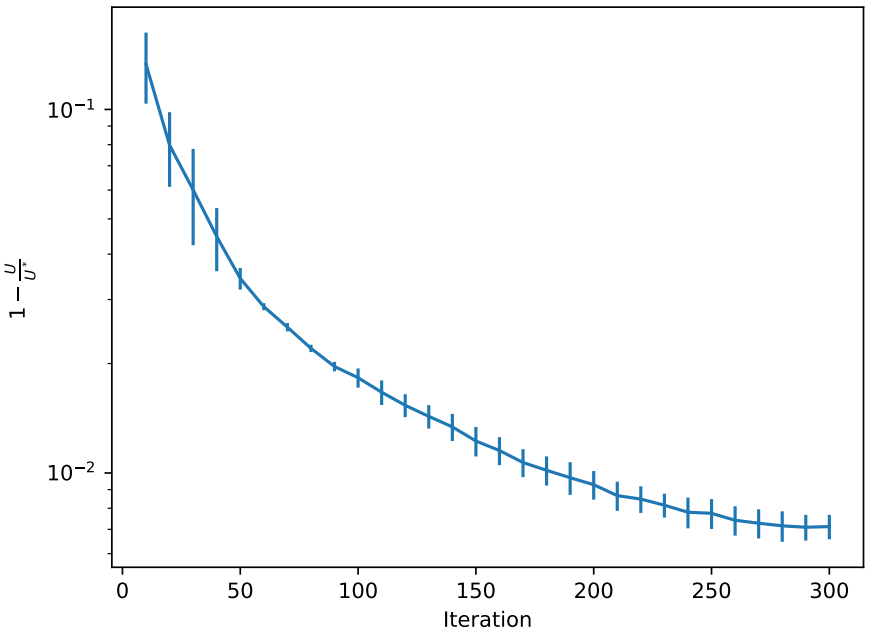


Figure A.3.: Lost utility by iteration for the best response learning experiment with $T = 4$ and fine-tuned parameters. Error bars indicate training uncertainty.

with $\delta_t = r_t + \gamma V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)$ being the temporal difference error.

This estimate is sensible if $V_\pi(s)$ captures the value of a state reasonability well. However, if demand is not observed, the only information to judge the value of a state on is the round index. Consider the extreme case where $\lambda = 0$. Then it holds that [14, Equation17]:

$$\hat{A}_t^{\mathrm{GAE}(0)} = \delta_t = r_t + \gamma V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t) \tag{A.17}$$

The value estimate of the state is independent of the action in this extreme case. Therefore, in the policy update, maximizing the advantage estimate is equivalent to maximizing the reward r_t, ignoring future rewards. Figure A.5 illustrates that this indeed happens; the myopic strategy is learned for $\lambda = 0$. It is only for $\lambda = 1$ that the advantage estimate is unbiased and the strategy converges to the best response. This information is valuable for the symmetric NE learning experiment, where no best response is available to compare to, and the demand is not observed.

Even with adequately-configured GAE-parameters we observed the training to perform significantly better if demand is observed, especially if the initial demand is randomized, as depicted in Figure A.6. This makes sense since demand observation allows the value function to do its job of predicting the value of a state.

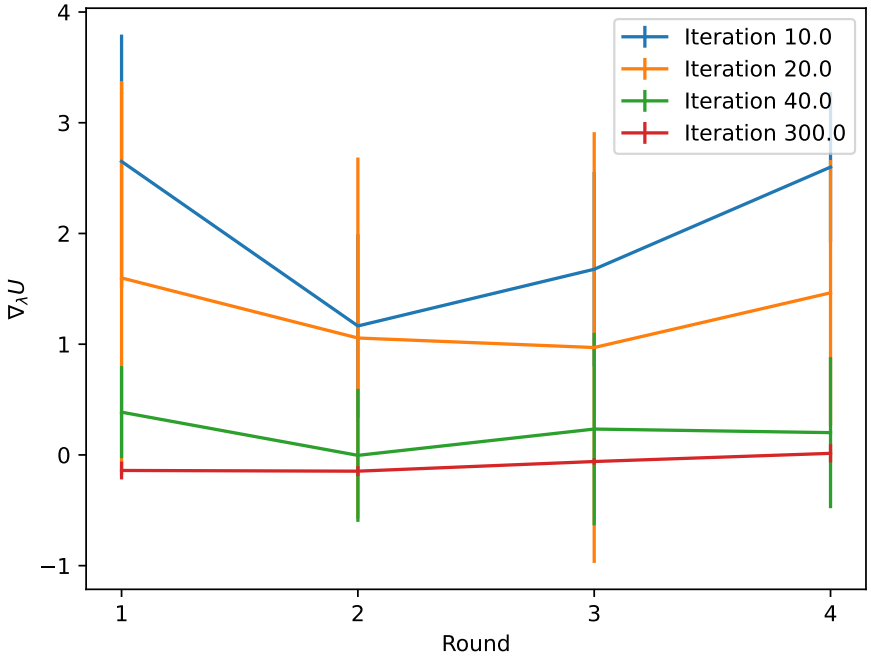


Figure A.4.: Numerical gradients of the utility w.r.t. the policy means for the best response learning experiment with $T = 4$ and fine-tuned parameters. Gradients are estimated by measuring the policy mean and variance as learned by the agent, and computing the change in utility when slightly altering the means. Error bars indicate training uncertainty.

A.3.3. GAE in the symmetric NE learning

While trying to learn the symmetric NE, it was observed that the GAE-lambda has an enormous influence on the learning outcome, too. For the cases without demand observation, it is plausible for this to be the same issue as described above. However, the influence is still present with demand observation (refer to Figure A.7), which we currently cannot explain. We picked $\lambda = 0.95$ as a reasonable default value. $\lambda = 0.9$ gives a much better fit with the best response but conveys a false sense of convergence.

A.3.4. Testing the verifier

To test the functionality of our brute force verifier (see Section 3.4) we created a best-response-learning-scenario where the agent is technically unable to learn the analytical best response: We randomized the initial demand, parameterized the action via the

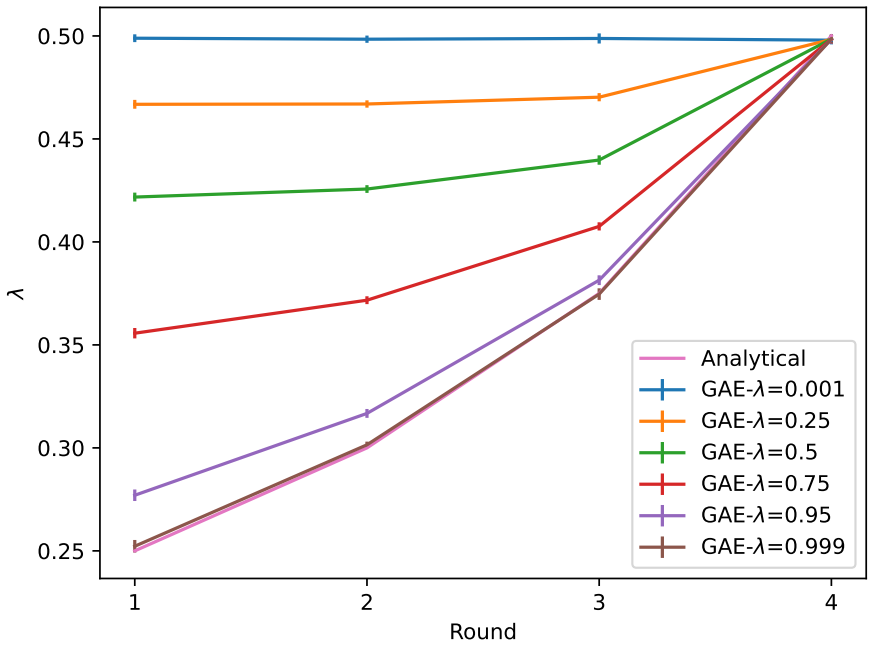


Figure A.5.: Strategy learned in the best response learning experiment with $T = 4$ for different values of the GAE-λ.

absolute price, and did not allow demand observation. Not knowing its demand, it is impossible to set an absolute price corresponding to the optimal price indicator. As expected, we observed that the utility loss compared to the best response is significant, but the loss estimated by the verifier remains small, as shown in Figure A.8. Further, it can be noticed that in the worst case, the discretization error may account for up to 10% of the utility loss.

A.3.5. Dynamic oligopoly parameter values

The parameters of the asymmetric experiments may seem arbitrary at first glance. This is true for the orders of magnitude of the parameters, the effect of this is still to be investigated. However, their relative positioning as an underlying thought. We want to observe the effects of agents dropping out of the game and a monopoly being created. To do so, we incentivized pushing agents out of the market as much as possible: We chose an initial demand close to the cost value ($c = 4$, $D = 5$). This both ensures that it is easy to push an agent out of the market (low relative demand potential to overcome) and that the remaining agents profit significantly from pushing the other agent out.

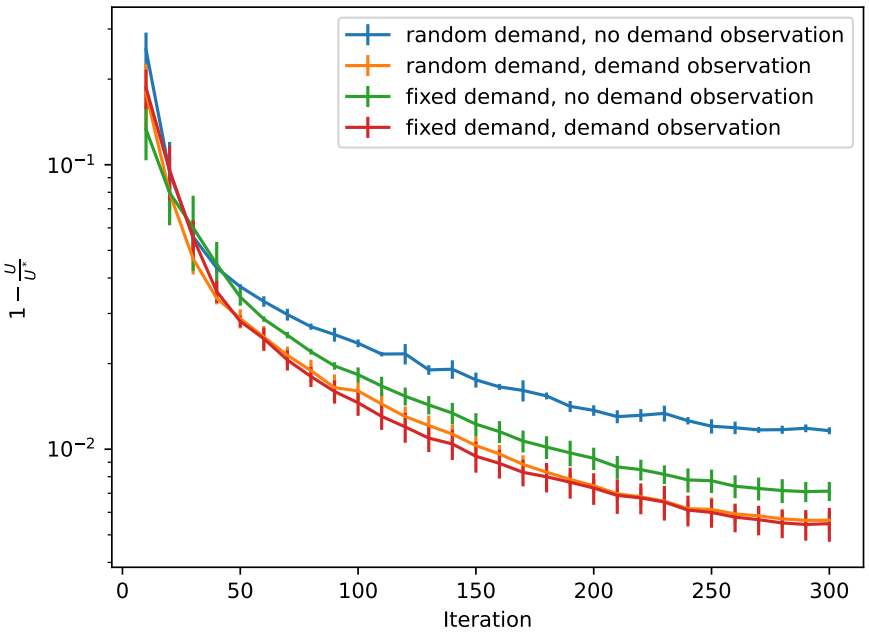


Figure A.6.: Relative utility loss with and without demand observation for the best response learning experiment with $T = 4$, with and without randomized initial demand, GAE-$\lambda = 0.999$. Error bars indicate training uncertainty.

The number of rounds was chosen high enough for the effect to happen.

A.3.6. Clamping the action space

A special case to be dealt with is the action parameterization p. Unlike in the case of the λ parameterization, the interval of valid pricing actions depends on the state of the game since both costs and demand are state-dependent. As a first step, actions are clamped to the valid interval by the environment. However, with this fix comes another problem: With the initial weights, the RL algorithms sample actions from a normal distribution with a mean of zero and configurable standard deviation. Best results are expected to be achieved by very small standard deviations, e.g. $\sigma = 0.01$. As a result, sampling actions greater than costs of e.g. $c = 3$ is vastly unlikely, leading the price to be clamped to the lower bound of the valid interval almost all the time. Consequently, the agents' actions do not affect the course of the game, and the RL algorithm cannot learn anything.

To solve this problem, the actions sampled by the RL algorithms are shifted by the minimum possible costs of the agent, i.e. the agent's costs if costs are fixed or the

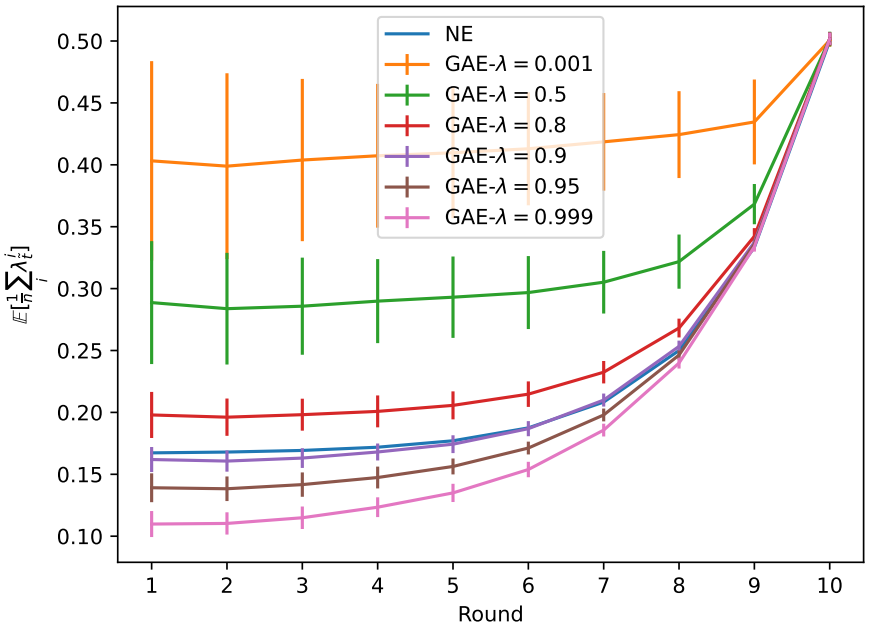
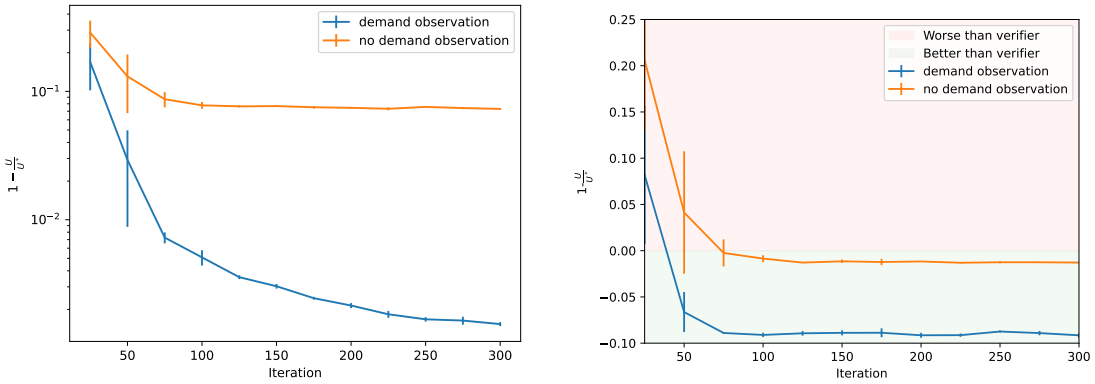


Figure A.7.: Results of the symmetric NE learning experiment with $T = 10$ with demand observation for different values of the GAE-λ. The price indicators have been averaged over the agents. The error bars are training uncertainties.



(a) Analytical response

(b) Verifier estimate

Figure A.8.: Utility loss estimated by the analytical deviation and the brute force verifier. The action is parameterized by the price and the initial demand is randomized. Without demand observation, it is impossible to reach the utility of the analytical best response.

minimum value of the probability distribution if costs are sampled. This way, the RL algorithm's initial actions have a non-negligible probability of being valid, avoiding the problem described above. Still, the agent gets no additional information about the state of the game from the shift because the amount of the shift solely depends on the configuration of the environment, not on the state of the game. This matters since, for instance, to play the λ parameterization in the real world, an agent would have to know their costs and demand, which may not be realistic.

A.3.7. PPO network architecture

For all experiments, we use the same actor-critic network, as illustrated in Figure A.9. The policy network and the value network consist of two fully connected layers with 64 neurons each, giving the network sufficient capacity to capture the dynamics of the game. The output is a normal distribution where the mean is the output of the policy network and the variance is a parameter, therefore the variance does not depend on the observation.

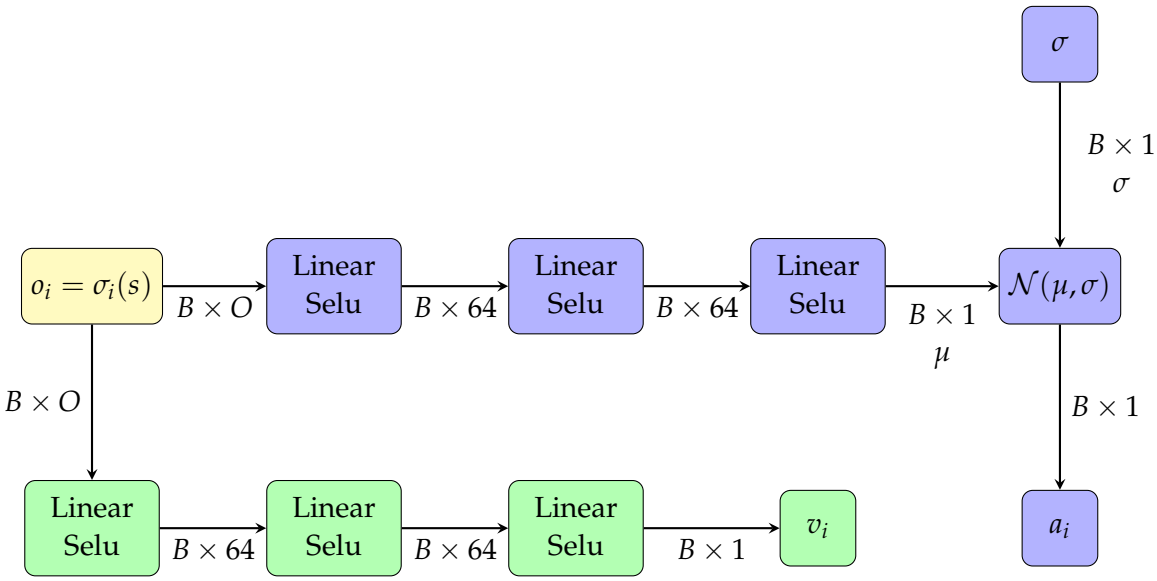


Figure A.9.: Architecture for the PPO policy and value networks. The observations o_i have a batch size of B and consist of B scalar values each. The policy network predicts the mean of a normal distribution of which the variance is a parameter. The action a_i is sampled from that normal distribution. The value network predicts the value v_i of the state.

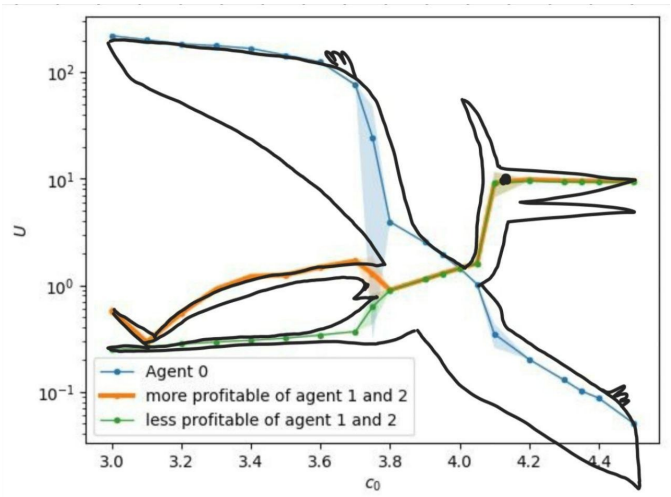


Figure A.10.: Artistic impression of the large asymmetric NE learning experiment result (see Figure 6.5) by Moritz Barth.