STRATEGIES AND EQUILIBRIA ON SELECTED MARKETS: A MULTI-AGENT SIMULATION AND STOCHASTIC MODELING APPROACH

ALDRIC LABARTHE, SUPERVISED BY JULIEN RANDON-FURLING

ABSTRACT. In this paper, we introduce the first agent-based model of competition in quantities featuring a *Deep Deterministic Policy Gradient* (DDPG) algorithm. This algorithm has been selected as a replacement for the traditional Q-Learning algorithm to examine two current unsolved questions in the economic literature: the tendency of algorithmic markets to converge toward a collusive equilibrium, and the chaotic behavior of the dynamic Cournot oligopoly. We show that the DDPG algorithm is a relevant tool to model oligopolies with independent learning agents. We find that our model consistently converges toward the Nash-equilibrium in every market structure we have tested, except for the Cournot oligopoly with well-tuned parameters. We estimate the effect of these parameters on the decision process and explain why collusion may occur in this situation. Overall, we show that algorithmic collusion remains an exception when algorithmic complexity increases. We also place our model in chaotic settings and find that the chaotic behavior of the dynamic Cournot model was only theoretical and never observed in simulations.

Keywords: *agent-based computational economics*, *algorithmic collusion*, *dynamic Cournot chaotic equilibrium*, *Deep Deterministic Policy gradient*, *algorithmic collusion*, *reinforcement learning*
JEL Classification Codes: C61, C62, C63, C73, D21, D43, D83, L13, L40

école
normale
supérieure
paris—saclay

ECOLE NORMALE SUPÉRIEURE (ENS) PARIS-SACLAY, GIF-SUR-YVETTE, 91190 FRANCE
E-mail address: aldric.labarthe@ens-paris-saclay.fr.
Date: 2024.

Contents

Introduction

Over the last few decades, the exponential rise of computational abilities and information availability has paved the way for new practices in pricing methods for firms to emerge. The use of algorithmic pricing, i.e. the set of technics that consists of using an algorithm to automatically adjust prices, has become more and more widespread, transitioning from being a digital marketplace specificity (Chen, Mislove, and Wilson 2016) to become a common tool used in almost every industry. This dramatic shift in pricing technics have raised regulatory concerns (OECD 2017, Competition and Markets Authority 2021) about possible effects of algorithms on selected prices that could lead to spontaneous collusion, i. e. a situation where competitors are spontaneously selecting prices that increase their profit beyond what should competition bring them, without necessarily forming a cartel with communication.

In the meantime, algorithmic pricing and more generally breakthroughs in self-learning decision algorithms are an opportunity for theoretical economists to reconsider classic questions and results. Hence, our motivation is plural. First, we are writing in the continuation of the works of the Agent-based computational economics movement, especially we embrace the principles of agent-based modeling defined by Tesfatsion 2017; Tesfatsion 2023 (MP axioms: agent's definition, scope, local constructivity, autonomy; system constructivity and historicity). This new way of conducting economic studies appears well suited to our topic: by removing the need of obtaining analytical solutions and results, we are free to implement as many refinements and heterogeneities as we want in our way of building agents, and, thus may access results and market configurations that were previously unreachable due to their mathematical complexity.

Then, we are writing in reaction to the ground-breaking results of several papers from Waltman and Kaymak 2008 to Calvano, Calzolari, and Denicolò 2019 on the effects of algorithmic competition on market price and collusion. These pioneering works with Q-Learning agents are nonetheless based on technics that seem outdated and partially inaccurate, justifying the need of further improvements and works to challenge the first insights they have provided. In the meantime, as the literature on AI and especially the reinforcement learning (RL) literature flourishes, algorithmic possibilities are getting wider and wider, and the need of conducting similar investigations with modern and more advanced methods becomes more and more pressant, especially as some recent developments find contradictory results (Abada, Lambin, and Tchakarov 2022).

Last, but not least, we want to bring together the former fields of reinforcement learning and algorithmic competition, with the emergent subfield of chaos control of equilibria in economics. These interesting features of chaotic behavior and instability of equilibria in oligopolies are well studied in recent works in standard models (for instance in the model from Cournot 1838, Puu 2008, Lampart, Lampartová, and Orlando 2022, Agiza and Elsadany 2003, Agiza and Elsadany 2004), but seem quite overlooked in the algorithmic competition literature. Our aim here will be also to check whether these analytical properties can be transferred to our algorithmic markets.

Our study will be conducted in 5 parts. In **Section 1**, we propose a literature review on the two economics fields of algorithmic collusion and equilibrium stability. Then, in **Section 2** we will define the theoretical foundations of the tools that we will extract from the various fields that we have formerly introduced: game theory, microeconomics, reinforcement learning, agent-based modeling and chaos control. In **Section 3**, we will introduce our simulation methodology and our hyper-parameter calibration. In **Section 4**, we will highlight the main results of our simulations, in each type of oligopoly structure that we have decided to study. In **Section 5**, we will discuss our results and see how they relate to the current literature.

1. Literature review

We try in this section to give an overview of the state of the art for economics works that study algorithmic collusion and the stability of oligopolies equilibria. As they are two fully separate subfields, we will introduce them in two different subsections. Algorithmic choices and mathematical implementations are introduced and questioned with the relevant literature in the next section.

1.1. ***Algorithmic collusion***

The literature about algorithmic collusion studies the possibility that independent agents, modeled as independent learners, could learn not to play the Nash-equilibrium, i.e. what it is rational to play in a non-communication static game, and to play an action that is closer to what they should play if they were communicating and colluding. The literature is essentially populated by papers that use a Bertrand oligopoly (competition by prices), but some approaches (including ours) use the true Cournot model which is in quantities. This difference has no interpretative consequences, one should only keep in mind that colluding in prices is to choose a higher price, whereas colluding in quantities is to choose a lower quantity (cf. **Section 2**).

The seminal paper of the field is the work of Waltman and Kaymak 2008 that was, one of the firsts, or the first one to prove that Q-Learning independent agents can converge collectively toward an equilibrium that is deviating from the Nash equilibrium in the direction of the collusive equilibrium. This work has been replicated many times, such that it is impossible to make an exhaustive list of derived works. We pinpoint the famous work of Calvano, Calzolari, and Denicolò 2019 as it was one of the first to demonstrate that, not only Q-Learning agents were learning to collude, but they were also exhibiting punishment behaviors, which are characteristics of a collusive state. This demonstration has been made possible by the use of memory, and the departure from a group of studies that used state-less algorithms (we can mention the work of Asker, Fershtman, and Pakes 2022). Some very recent works have tried to give an analytical explanation for collusive outcomes, using the relative simplicity of Q-Learning. We can pinpoint the works of Banchio et al. 2022 and of Kerzreho 2024 that both use fluid approximations and differential equations to determine causes for spontaneous coupling, enlightening that Q-Learning algorithms overestimate the cumulative rewards associated with the states attainable after cooperative actions.

At the same time, empirical studies have been conducted and support the link between algorithmic-pricing and collusion. As detailed by Calvano, Calzolari, and Denicolò 2019, it is quite difficult to estimate empirically collusion, as collusion is prohibited in most countries. Nonetheless, several studies have been made and support the idea that firms can collude without necessarily forming a cartel with communication (we pinpoint the work of Byrne and Roos 2019), an a very recent attempt has proved a link between algorithmic pricing and collusive outcome by studying the case of Germany's retail gasoline market (Assad et al. 2024).

However, some very recent works have come to challenge these widely accepted results. Indeed, the former almost all rely on the same technology: Q-Learning (or even simpler algorithms), which is widely accepted as an outdated algorithm to solve continuous problems with large action spaces (we further explain this argument in **Section 2**). Abada, Lambin, and Tchakarov 2022 state that over-simplified algorithms like Q-Learning, or not well-tuned exploration processes, could be the source of these strange results.

1.2. ***Stability and chaos in oligopolies equilibria***

The subfield of chaotic equilibria is very old but has been quite overlooked for decades. The seminal paper is Theocharis 1960, which mathematically demonstrated that the Cournot "adjustment mechanism" (i.e. the mechanism by which the firms settle to the Nash equilibrium was first described without formalism by Cournot 1838, who postulated that this adjustment will lead toward a stable quantity) was in fact no guarantee for the equilibrium to be stable. Several papers have since demonstrated similar results, exploring different demand structures and costs functions, as Puu 2008, Agiza and Elsadany 2003, Agiza and Elsadany 2004.

While some further developments have been made with exotic evolutionary approaches (Hommes, Ochea, and Tuinstra 2011), the issue of the instability of the Cournot oligopoly remains an unsolved question. Several authors might have understood the mathematical implications of this issue, but no economical explanation has been suggested, let alone proved, to the best of our knowledge.

2. Theoretical Framework

2.1. ***Microeconomic and Game theoretic framework***

The main focus of this study is competition in quantities. We deliberately do not consider all market structures that leave the choice of a price to the firms. We hence focus on four market structures highly classic in economics: monopoly, Cournot duopoly, Stackelberg duopoly, Cournot oligopoly, and perfect competition. The first and the latter will not be studied for themselves but will be seen as limit cases of the two oligopolies. In this section, we will recall the main features of the models, and compute the analytical solutions in our simplified case.

For now on, we will consider a linear demand function which is defined by $D(p) = D - p$ and respectively an inverse demand function $D(Q) = D - Q$, with $D \in \mathbb{R}_+$ a parameter that will be studied later (and $p \in \mathbb{R}_+$ the price, and $Q \in \mathbb{R}_+$ the total production quantity). We use as notations q_i the quantity produced by firm i and $q_{-i} = Q - q_i$ the quantity produced by every other firms. We assume that each firm will share in every models a common quadratic cost function defined as $C_i(q_i) = cq_i^2$, with $c \in \mathbb{R}_+$.

2.1.1. *Stackelberg duopoly*

In this section, we consider $N = 2$ firms, a leader and a follower, perfectly independent firms without any communication channel, and which have a market power, that is their decisions can directly impact the market price. The game is assumed to be in perfect and complete information, and to be static. The leader decides first the production quantity according to his knowledge on the best response function of the follower, and the latter decides what to produce. We denote the variables of the follower by the index f and the ones of the leader by the index l. To solve this game, we use backward induction by starting with the second player:

$$\underset{q_f}{\operatorname{argmax}}\, \Pi_f(q_f, q_l) = (D - q_f - q_l)q_f - cq_f^2 \tag{2.1.1}$$

From which we can write the First Order Conditions (FOC)[1] and obtain the reaction function:

$$\frac{\partial \Pi_f(q_f, q_l)}{\partial q_f} = 0 \Leftrightarrow q_f^*(q_l) = \frac{D - q_l}{2(1+c)} \tag{2.1.2}$$

The leader, knowing the optimal behavior of the follower, decides its production accordingly:

$$\underset{q_l}{\operatorname{argmax}} \Pi_l(q_l, q_f) = \underset{q_l}{\operatorname{argmax}} \{\Pi_l(q_l, q_f^*(q_l)) = (D - q_l - \frac{D - q_l}{2(1+c)})q_l - cq_l^2\} \tag{2.1.3}$$

Following the same steps as for 2.1.2, we have:

$$q_f^* = \frac{D(1+6c+2c^2)}{4(1+c)(1+4c+c^2)} \quad q_l^* = \frac{D(1+2c)}{2+8c+c^2} \tag{2.1.4}$$

2.1.2. *Cournot oligopoly*

In this section we consider $N \geqslant 2$ perfectly independent firms without any communication channel, and which have a market power, that is their decisions can directly impact the market price. First, we will consider a simplified case, with perfect and complete information. Each firm perfectly knows the cost structure of each other firm and can perfectly determine the best response function of every other firm. Moreover, it is assumed that the game is static.

For each firm i, we can write the following profit function (objective function):

$$\underset{q_i}{\operatorname{argmax}} \Pi_i(q_i, q_{-i}) = (D - q_i - q_{-i})q_i - cq_i^2 \tag{2.1.5}$$

From which we can write the First Order Conditions (FOC) and obtain the reaction function:

$$\frac{\partial \Pi_i(q_i, q_{-i})}{\partial q_i} = 0 \Leftrightarrow q_i(q_{-i}) = \frac{D - q_{-i}}{2(1+c)} \tag{2.1.6}$$

As we assumed perfect and complete information, the firm knows the reaction function of its competitors and can therefore replace it in 2.1.6 to find:

$$q_i^* = \frac{D}{(N+1)+2c} \tag{2.1.7}$$

One famous result is that, as firms are perfectly identical, their quantity is the same.

2.1.3. *Dynamic Cournot model*

In this study, we choose to introduce a more complex model as the previous ones. Indeed, the two previous ones suffer from two limits: first, they heavily rely on hypothesis (complete and perfect information) and they do not incorporate time. As we want to use self-learning algorithms, the process by which the equilibrium settles, is as interesting to us as the equilibrium itself. Hence, we introduce what we call a *dynamic Cournot model* - as several authors after Cournot have tend to prefer it, but its diffusion seems to have been very curtained in the previous decades without obvious reasons. We first partially release the perfect information hypothesis by setting that firms no longer know the production technology of their competitors, such that they are not able to compute their reaction functions. Moreover, they cannot observe the quantities produced in the current period before selecting their actions and hence can only base their decision on the last total played quantities in the previous period.

[1]For the sake of clarity, we assume Second order conditions validated, which is obvious in most of our maximisations as our demands are linear and our cost quadratic, implying a strictly concave profit.

Hence, the profit of firm i becomes:

$$\underset{q_t^i}{\operatorname{argmax}}\, \Pi_t^i(q_t^i, q_{t-1}^{-i}) = (D - q_t^i - q_{t-1}^{-i})q_t^i - c(q_t^i)^2 \tag{2.1.8}$$

As before we write the First Order Conditions (FOC) and get the reaction function:

$$\frac{\partial \Pi_t^i(q_t^i, q_{t-1}^{-i})}{\partial q_t^i} = 0 \Leftrightarrow q_t^i(q_{t-1}^{-i}) = \frac{D - q_{t-1}^{-i}}{2(1+c)} \tag{2.1.9}$$

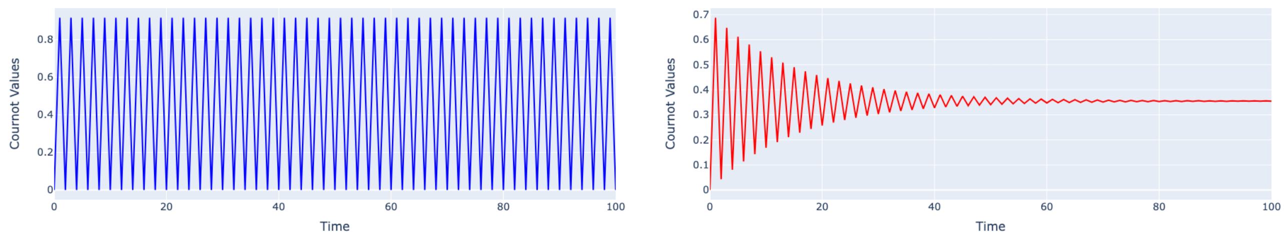
Current produced quantities are determined by the following system of first-order difference equations:

$$\begin{cases} q_t^1 = \frac{D - \sum_{i\neq 1} q_{t-1}^i}{2(1+c)} \\ \vdots \\ q_t^N = \frac{D - \sum_{i\neq N} q_{t-1}^i}{2(1+c)} \end{cases} \Leftrightarrow \underbrace{\begin{bmatrix} q_t^1 \\ \vdots \\ q_t^N \end{bmatrix}}_{Q_t} = \frac{-1}{2(1+c)} \left(\underbrace{\begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \ddots & \vdots \\ 1 & 1 & \ddots & \ddots & 1 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} q_{t-1}^1 \\ \vdots \\ q_{t-1}^N \end{bmatrix}}_{Q_{t-1}} - \begin{bmatrix} D \\ \vdots \\ D \end{bmatrix} \right) \tag{2.1.10}$$

Two observations must be made: first, the steady-state of these first-order difference equations is the classical Cournot oligopoly that we presented before (2.1.7, cf. Appendix 1 for a detailed proof). Secondly, as this system is a differential system, we need to check for stability. We therefore analyse the eigenvalues of the squared matrix A:

$$\begin{aligned} &\mathrm{Sp}(A) = \{-1, (N-1)\} \\ &\Rightarrow r(A) = \max(|\mathrm{Sp}(A)|) = |N-1| \\ &\Rightarrow r\left(\frac{-1}{2(1+c)}A\right) \leqslant 1 \Leftrightarrow c \geqslant \frac{N-3}{2} \end{aligned} \tag{2.1.11}$$

This stability condition (*which is developed and proved in Appendix A*) will later be very important when c will be fixed. For now, we can say that the oligopoly is trivially stable for $N \leqslant 3$ which is a standard result obtained by Theocharis 1960. When the stability condition is met, the system converges toward the steady-state, i. e. the normal case. In the contrary, when the condition is not met, the system diverges, although when fixing bounds for quantities, the system tends to alternate between the bounds of the set for q_t^i.



(A) $c = 0.2$ (B) $c = 0.55$

FIGURE 2.1.1. **Numerical simulations illustrating the behavior of the Cournot system of quantities** (simulations conducted with $D = 2.2$ and $N = 4$, *quantities are bounded between 0 and 1*).

2.2. ***Reinforcement learning and agent-based modeling***

2.2.1. *Agent-based modeling*

Contrary to the standard way of studying oligopolies in the economic literature, we choose to implement an agent-based model. Indeed, our objective is double: first, we want to develop a model free approach without almost any assumptions given to agents. We model agents as fully autonomous without any information on other agents nor on the market structure. Their only property and certainty is that their objective is to maximize their profit. This framework allows us to study cases with imperfect or incomplete information, and to avoid making unrealistic assumptions. Secondly, as we discovered with the dynamical Cournot model, the aim of this study is also to explore not well-defined equilibria, where analytical solutions are not available. Agent-based simulations are one of the few available and reliable tools that can allow us to study this type of phenomena.

One other interesting advantage of this modeling approach is that it allows us to study decentralized equilibria that are more realistic. Indeed, as mentioned in the introduction, one of the main driving factor of the growing interest for artificial markets is the surge in the use of algorithmic pricing. Using an agent-based model is not only the sole option to implement such algorithms, but also the most realistic as it reproduces quite well the true mechanism of the markets of algorithms. Nonetheless, the only limitation on this matter is that we will focus on synchronous learning, which is not a realistic assumption in most cases. A natural extension of this work will be to consider the asynchronous case.

To be precise, we consider each firm to be a separate fully autonomous agent. Consumers are not modeled here: we consider only a representative consumer from which comes all the demand. The consumer is not modeled as an agent: it is more like a market clearer with no autonomy. Here, we suppose that no information is shared among firms, and no information on the demand is provided. Hence, for instance, our agents are not implemented with some knowledge on the reaction function of the other firm. Agents are totally isolated: each firm chooses its level of production, and receives from the market clearer the price at which it has been bought.

2.2.2. *Reinforcement learning*

As agents are fully separated and autonomous, they need to be able to elaborate strategies and to react to the responses of the environment. We decide to have a model-free approach: no strategies are preregistered (hardcoded) in the simulations, and agents are free to behave as they want. Due to the structure environment-agent that we previously described, reinforcement learning is the standard tool used to implement the decision process of agents.

The classical tool which was widely used is Q-Learning (proposed by Watkins 1989 and famously used by Calvano, Calzolari, and Denicolò 2019). We consider a Markov-game (Littman 1994) composed of $N \subset \mathbb{N}$ the set of agents (for simplicity we consider $\text{card}(N) = N \in \mathbb{N}$), $\mathcal{A}_{j \in N} \subset \mathbb{R}$ (finite and countable) the set of actions available for each agent, S the set of all possible states, and the markovian transition function $p : S \times \bigtimes_{j \in N} \mathcal{A}_j \longrightarrow S$ such that $s_{t+1} = p(s_t, a_1, ..., a_N)$. We want to highlight here the fact that we are in the framework of a Markov game, allowing us to only consider the previous state and the current actions of players to deduce the next state.

Each firm (hereinafter agent) receives after each action (and hence state transition) the reward r computed by the mapping $r : S \times \mathcal{A}_j \longrightarrow \mathbb{R}$. Contrary to the analytical way of modeling economics, the agent here does not maximize its profit but its expected total reward:

$R = \sum_t^{\infty} \gamma^t r(s_t, a_t)$ (with $\gamma \in [0,1]$ a discount factor). We can write the program of the agent (known as the Bellman's equation) as $V(s) = \max_{a \in \mathcal{A}} \{\mathbb{E}[r(s,a)|\ s,a] + \gamma \mathbb{E}[V(s')|\ s,a]\}$. We can rewrite it and obtain the classical equation of Q-learning:

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

Many issues arise from this technic. First, $\mathcal{A}_j \subset \mathbb{R}$ is a finite and countable set which is not well suited to the task we are trying to model: prices are by definition continuous, and the discretization of this action space has not been yet proved effect-less on the final results. Worst, as Q-learning relies on the computation of a Q-Matrix (a table for $Q(s,a)$ values, of dimension $\text{card}(S) \times \text{card}(A)$), it fails to scale when the dimension of the action space increases. Some developments (the most famous is Mnih et al. 2013) have tried to improve Q-Learning with for instance the use of neural networks as approximations for the Q function (Deep Q-learning). Nonetheless, these methods still struggle with continuous and high-dimensional action spaces.

The use of these technics (Q-learning and its variants) has become more and more widespread in the economic literature despite these limitations due to their relative simplicity (especially for the implementation part). However, if one could bear to make assumptions to accept the use of these technics, it seems quite difficult to accept without question the results that they allow to discover when they go against all that the standard theory produce: indeed, we can highlight the famous work of Calvano, Calzolari, and Denicolò 2019 which prove that Q-learning agents in a Cournot oligopoly learn tacit collusion instead of converging toward the Nash equilibrium, as microeconomics would predict. These results appear difficult to fully embrace as they heavily rely on an algorithm that is not well-suited for the task we are using it to. In order to challenge these conclusions and not to fall in the same perils than the previous works, we choose to use a more modern approach directly extracted from the reinforcement learning field, which is still very little used in the economic literature.

2.2.3. *A Deep Deterministic policy gradient algorithm*

The algorithm we have chosen to implement is directly derived from the work of Lillicrap et al. 2015, which itself heavily relies on Silver et al. 2014 and Konda and Tsitsiklis 1999 (the seminal paper who introduced Actor Critic algorithms). The use of this type of algorithm is still rare in the economic literature: we can identify the works of Abada, Lambin, and Tchakarov 2022 who used an Actor-critic algorithm and Graf et al. 2023. We stay in the same framework as introduced for the Q-learning section, with the same transition and reward functions. Our algorithm follows an actor-critic approach: first, the agent learns a policy π which is a mapping $\pi : S \longrightarrow \mathcal{P}(\mathcal{A})$ where $\mathcal{P}(\mathcal{A})$ is the set of probability measures on the action space. Hereinafter, we will be dealing with deterministic policies and not stochastic policies. Hence, π can be fully described by the deterministic mapping:

$$\begin{aligned} \mu : S &\longrightarrow \mathcal{A} \\ s &\longmapsto \text{argmax}_a Q^{\pi}(s,a) \end{aligned}$$

Then, the agent has to estimate the reward function ($Q^{\pi}(s,a)$, the critic part of our actor-critic algorithm). As, it uses his policy to interact with the environment (its actions follow a probability distribution), and perceives a gain accordingly, its objectif function can be rewritten as:

$$J^{\pi} = \mathbb{E}_{s_i \sim E, a_i \sim \pi}[R|s_0, a_0]$$

We can also write the expression of the action-value function, to follow a similar framework as Q-learning:

$$\begin{aligned} Q^\pi(s_t, a_t^i) &= \mathbb{E}_{s_i\sim E, a_t^i\sim\pi}[R_t|s_t, a_t^i] \\ &= \mathbb{E}_{s_{t+1}\sim E}[r(s_t, a_t^i) + \gamma \mathbb{E}_{a_{t+1}^i\sim\pi}[Q^\pi(s_{t+1}, a_{t+1}^i)]|s_t, a_t^i] \end{aligned}$$

As the policy is fully deterministic, we can simplify:

$$Q^\pi(s_t, a_t^i) = \mathbb{E}_{s_{t+1}\sim E}[r(s_t, a_t^i) + \gamma Q^\pi(s_{t+1}, \mu(s_{t+1}))|s_t, a_t^i] \tag{2.2.1}$$

Now that actor and critic functions have been fully described, we need to highlight how they are approximated. Indeed, the goal of the algorithm is to make two convergent estimators of μ and Q^π. We use neural networks as estimators, and consider $\mu_{\theta_\pi} = \mu(s|\theta_\pi)$ as our estimator of μ with a vector of parameter θ_π (weights and biases of the neural network), and $Q^\pi_{\theta_Q} = Q^\pi(s, a|\theta_Q)$ for Q^π with parameters θ_Q. Our estimators are constructed using a replay buffer ($\mathcal{B}$) as suggested by Lillicrap et al. 2015, where we store all previous experiences as tuples $(s_t, r_t^i, a_t^1, ..., a_t^N, s_{t+1})$.

To calibrate our critic estimator we define a loss function with respect to θ_Q, parameters that will be fitted using a gradient descent that we will later introduce:

$$\mathcal{L}(\theta_Q) = \mathbb{E}[(Q^\pi(s_t, a_t^i|\theta_Q) - \big(r(s_t, a_t^i) + \gamma Q^\pi(s_{t+1}, \mu(s_{t+1}))\big)^2 \,|r, a_t, s_t, s_{t+1}] \tag{2.2.2}$$

Which can be rewritten in a computable form, when evaluated on $B \subset \mathcal{B}$:

$$L(\theta_Q, B) = \frac{1}{\text{card}(B)} \sum_{(s_t, r_t^i, a_t^1, ..., a_t^N, s_{t+1})\in B} [Q^\pi(s_t, a_t^i|\theta_Q) - (r(s_t, a_t^i) + \gamma Q^\pi(s_{t+1}, \mu(s_{t+1})))]^2 \tag{2.2.3}$$

This calibration is actually very classic in the AI field, and does not raise any issue, as we somewhat know the value toward which our estimator should converge (we can compute the Q function in simplified cases, and we give to the agent a reward that is design to make them learn this function). Although, the actor case appears far more challenging: in this case we want an estimator of the argmax function of an estimator. Here, it is not possible to minimize a loss between correct values and estimated values as it is quite impossible to perform a dynamic maximization of Q at each iteration. To solve this issue, we use the *Deterministic policy gradient theorem* from Silver et al. 2014.

> **Deterministic policy gradient theorem:** Under conditions elucidated in the next subsection, we have:
>
> $$\nabla_{\theta_\pi} J^\pi = \int_{\mathcal{S}} \rho(s) \nabla_{\theta_\pi} \mu(s|\theta_\pi) \nabla_a Q^\pi(s, a|\theta_Q)|_{a=\mu(s|\theta_\pi)} ds \tag{2.2.4}$$
>
> with $\rho(s)$ a discounted state distribution factor made accordingly to our markovian transition function p.

Equation 2.2.4 enables us to perform a gradient ascent (as we are maximizing and not minimizing) on J^π and to get a convergent estimator $\mu(s|\theta_\pi)$, without performing any time-consuming dynamic maximization of Q. As observed Lillicrap et al. 2015, the factor $\rho(s)$ is ignored in the literature without significant alterations to the results.

Nonetheless, Lillicrap et al. 2015, the first paper to have implemented such algorithm, observes that there is a great instability when directly using eq. 2.2.4 on neural networks. Indeed, at each learning step, $\nabla_{\theta_\pi} J^\pi$, -that is used to fit μ_{θ_π}-, is computed using values from Q^π, which itself is currently being fitted by a gradient descent on $\nabla_{\theta_Q} L(\theta_Q, B)$, which is computed using $\mu_{\theta_\pi}(s_{t+1})$. In order to stabilize the learning process and avoid these circular references, we follow an approach initiated by Mnih et al. 2013 and introduce two other estimators (neural networks): $\mu_{\theta_\pi}^{\text{target}} = \mu^{\text{target}}(s|\theta_\pi)$ and $Q_{\theta_Q}^{\pi,\text{target}} = Q^{\pi,\text{target}}(s,a|\theta_Q)$. These target estimators are delayed versions of our primary estimators that are cloned with them every κ training steps. This cloning operation is a Poliak averaging:

$$\begin{cases} \theta_\pi^{\text{target}} = \tau\theta_\pi + (1-\tau)\theta_\pi^{\text{target}} \\ \theta_Q^{\text{target}} = \tau\theta_Q + (1-\tau)\theta_Q^{\text{target}} \end{cases} \quad \text{with } \tau \in [0,1] \tag{2.2.5}$$

Finally, we can rewrite our two main fitting equations (2.2.3, 2.2.4) with target networks:

$$L(\theta_Q, B) = \frac{1}{\text{card}(B)} \sum_{(s_t, r_t^i, a_t, s_{t+1}) \in B} \left[Q^\pi(s_t, a_t^i|\theta_Q) - (r(s_t, a_t^i) + \gamma Q^{\pi,\text{tar}}(s_{t+1}, \mu^{\text{tar}}(s_{t+1})))\right]^2 \tag{2.2.6}$$

$$\nabla_{\theta_\pi} J^\pi = \int_{\mathcal{S}} \nabla_{\theta_\pi}\mu(s|\theta_\pi)\nabla_a Q^\pi(s,a|\theta_Q)|_{a=\mu(s|\theta_\pi)} ds \tag{2.2.7}$$

As in every machine-learning algorithm, we implement an exploration policy $\mathcal{G}$. Here we decide to use a white noise, such that $a_t^i = \mu(s|\theta_\pi) + \mathcal{G}$ with $\mathcal{G} \hookrightarrow \mathcal{N}(0, \sigma(t))$ with $\sigma(t)$ a function of the time that will need to be elucidated.

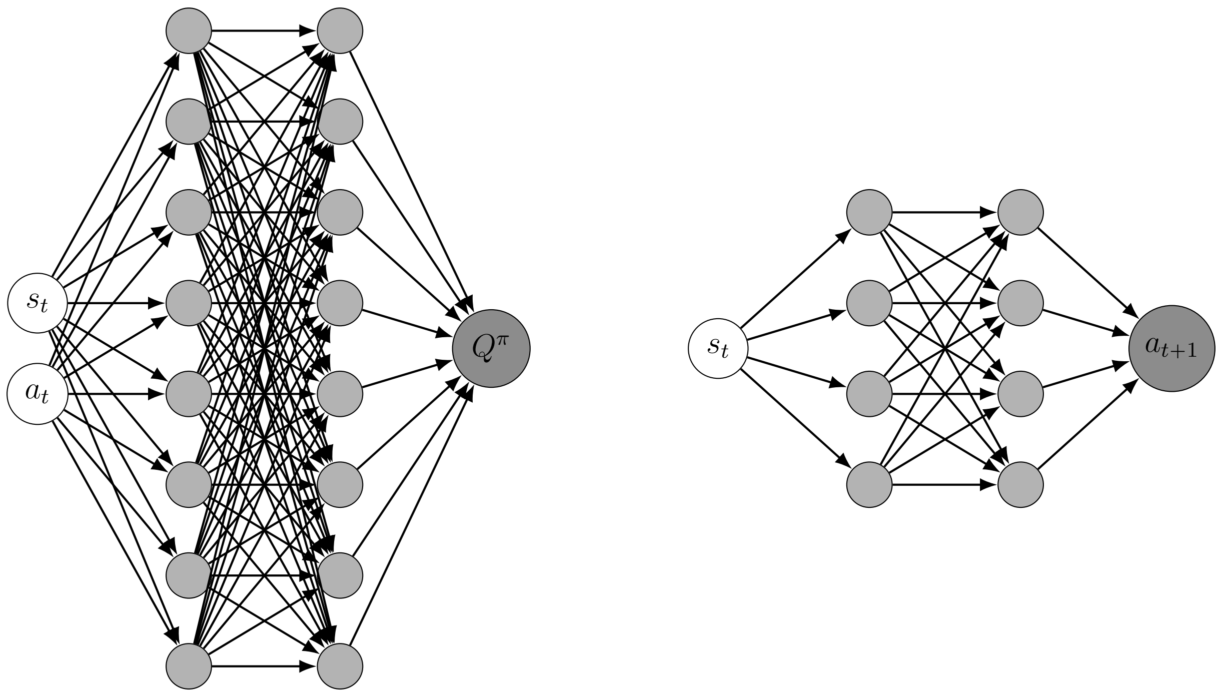


FIGURE 2.2.1. **Critic and Actor neural networks estimators.**
Both networks feature two dense fully connected hidden layers with ReLU activation and one output with sigmoid activation.

Algorithm 1 Deep Deterministic policy gradient (based on Lillicrap et al. 2015), for agent i

```
1: for $t \in [\![1, T]\!]$ do
2:     Observe state $s_t$
3:     Select action using $a_t = \mu(s|\theta_\pi) + \mathcal{G}$
4:     Observe reward $r_t^i$, next state $s_{t+1}$ and store in $\mathcal{B}$ $(s_t, r_t^i, a_t^1, ..., a_t^i, ..., a_t^N, s_{t+1})$
5:     for $b \in [\![1, \beta]\!]$ do
6:         Sample a set $B$ of transitions from $\mathcal{B}$
7:         for $e \in [\![1, \mathrm{card}(B)]\!]$ do
8:             Compute gradient of the Critic's error loss 2.2.6
9:             Compute gradient of the Actor part with the DPG Theorem 2.2.7
10:        end for
11:        Perform a gradient ascent on the mean of Actor gradients
12:        Perform a gradient descent on the mean of Critic gradients
13:        Update target networks with a Polyak average 2.2.5
14:    end for
15: end for
```

Memorandum on the key concepts underlying our DDPG implementation

- *Learning*: an algorithm for learning is typically a sequence of instructions for adapting the values of certain parameters in a model designed to describe a certain type of data. Think of simple linear regression in the univariate case ($Y = aX + b$), then computing the intercept (b) and the slope (a) of the regression according to, say, ordinary least squares on a given data set, is a procedure to "learn" a linear model of the data present in the data set. This set may then be considered as a training set, and the linear function that has been learnt (“trained”) on it can be used to make predictions on other data.

- *Exploration procedure, gradient descent/ascent*: a learning procedure, as described above, would usually involve an optimization, eg minimizing some function that is considered to represent an error in the prediction or a loss in the description of the data (hence often called "loss function", or "cost function"). When it is not possible to compute exactly the optimal point (in linear regression with ordinary least squares, it is), one has to resort to explore the set of all possible models, looking for the one that minimizes the loss function (or equivalently maximizes some function measuring the good quality of an approximation). Hence the need for an exploration strategy. Classic strategies include starting with a randomly chosen model and moving in a direction (i.e. changing the values of the parameters) where the value of the quality or the loss function increases or decreases -that is, following the direction or the opposite direction of the gradient.

- *Learning rate*: in a learning algorithm and/or an optimization procedure, there would usually be a parameter accounting for either the size of the step we take eg in the gradient descent or the importance we give to new data points compared to the ones already used to "learn" coefficients. Such a parameter is called the *learning rate*: if put to 0, nothing happens, the current values of the model parameters never change, the algorithm does not "learn" anything anymore.

- *Neural Networks*: just like in the case of regressions, these are functions (albeit, complicated sequences of functions composed with one another, especially in the case of so-called *deep learning*), that are designed to model data. They may be thought of as networks of virtual neurons that receive, process and output information. Neural networks usually have many more parameters than simple regressions, in particular the "weights" that encode the influence of neurons on one another.

- *Deep deterministic policy gradient algorithm*: the purpose of the algorithm is to learn without any previous knowledge a function that maps states and actions (and which is called a *policy*). In our economic context, states are previous actions from other players, and actions are selected quantities. This policy is formally learned by a neural network called *Actor*. This network is trained using another network, the *Critic*, which aims to estimates the outcome of each possible pair (state, action). Both networks are trained using a *Replay buffer*, a collection of previously observed (state, action) pairs. To avoid instability, both networks are cloned in *target networks*, which are simply lagged copy of themselves used to stabilize training.

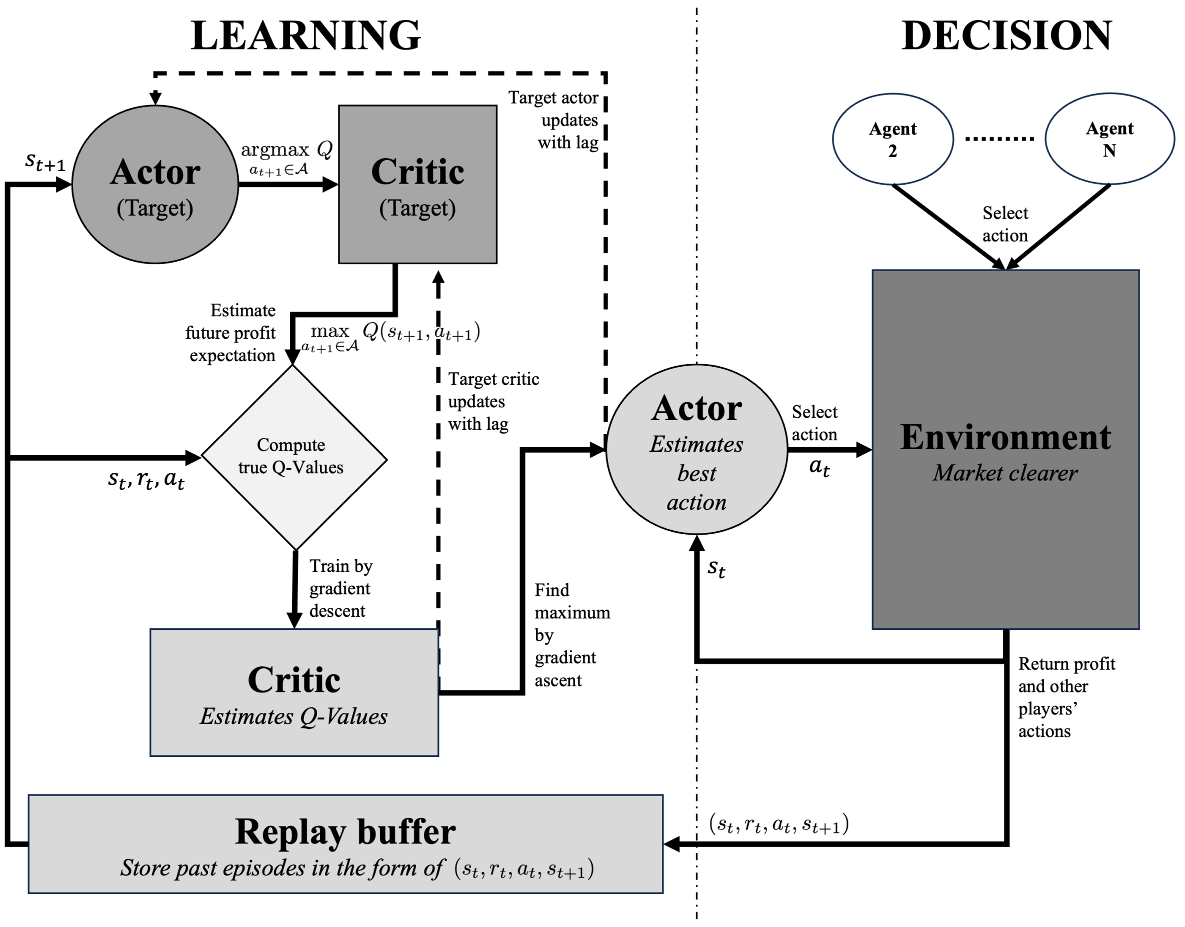

LEARNING
DECISION
Actor (Target)
Critic (Target)
$\underset{a_{t+1}\in\mathcal{A}}{\text{argmax}}\, Q$
s_{t+1}
Target actor updates with lag
Estimate future profit expectation
$\max_{a_{t+1}\in\mathcal{A}} Q(s_{t+1}, a_{t+1})$
Target critic updates with lag
Compute true Q-Values
s_t, r_t, a_t
Train by gradient descent
Critic
Estimates Q-Values
Find maximum by gradient ascent
Actor
Estimates best action
Agent 2
Agent N
Select action
Select action
a_t
Environment
Market clearer
s_t
Return profit and other players' actions
(s_t, r_t, a_t, s_{t+1})
Replay buffer
Store past episodes in the form of (s_t, r_t, a_t, s_{t+1})

FIGURE 2.2.2. **A summary of our algorithm design (we use a DDPG algorithm)**

2.3. *Markov decision games and convergence issue*

The previous framework in which we have laid the foundations of our reinforcement learning algorithm is a Markov game directly derived from the work of Littman 1994. Before using our algorithm in oligopolies described in our microeconomic theoretical framework, we need to discuss theoretical properties of such games. To summarize in a game theoretic language, our aim is that every independent player (agent) finds and learns an optimal policy (that is undominated), and, hence, collectively converge to a Nash equilibrium. Unfortunately, there is currently no proof nor clue that the precedent framework is well suited to perform such a task. We will in this subsection review several issues that we need to tackle, and what unresolved uncertainties are left for future research.

First, the obvious question is: is-there truly an optimal policy for each agent to learn? and is there a Nash equilibrium? These questions are studied by Littman 1994 in a theoretical way. Nonetheless, the answers in this paper are not fully convincing in our setting. Indeed, it is proven that there exists an optimal policy, and this policy can be found using the Minmax classic way of solving games in game theory. First, the proof of existence of an optimal policy is achieved using stochastic policies. Yet, algorithms that we have studied in the previous part are all *deterministic*, that is they can only learn and find fully deterministic policies. Secondly, as we are dealing with oligopolies, the minmax solving way is not satisfactory (this can be seen by reducing the problem to a classic problem of static game theory and comparing obtained equilibria).

Secondly, if proofs of convergence are available for algorithms that we have studied before in the simple agent case, no proof exists - to the best of our knowledge - that such algorithms will truly learn an optimal policy in a multi-agent framework. Indeed, the main issue highlighted by Lowe et al. 2017 is that the transition function p that we have previously introduced does not only depend on the action of one agent, but on the actions of every other agent. This can lead to high variance in the learning process due to the fact that each agent cannot isolate the effect of its action on the state transition and reward. Worst, the gradient descent/ascent could be lured: learning a suboptimal policy could give higher rewards if at the same time the action of other agents are modifying favorably the reward of the first.

Several algorithms derived from the DDPG algorithm (Lillicrap et al. 2015) have been since introduced to tackle the issue of multi-agent settings. We choose to highlight some, and briefly explain why they do not fit our requirements.

- *Multi-Agent Deep Deterministic Policy Gradient* (MADDPG) from Lowe et al. 2017: this algorithm may be the most instrumental contribution to the field of multi agent reinforcement learning. Their approach relies on the principle of *centralized critic - decentralized actor*, which is not well suited to our study. Indeed, it is highly non-realistic that competitors would make their algorithms train together and share the same centralized critic. Hence, despite being very promising in terms of performance, this technical solution is not fitted to our economic study.

- *Multi-Agent Actor-Critic with Networked Agents* from Zhang et al. 2018: this algorithm features fully decentralized agents but is only designed to work for cooperative games where agents have to work to optimize the global reward. Again, such technics would be in reality fully prohibited by law in the context of competition.

- *Distributed Deep Deterministic Policy Gradient* (3DPG) from Redder, Ramaswamy, and Karl 2022: this algorithm is very interesting as it comes with a convergence proof.

Yet, this proof relies on the use, not of the actions of other agents, but directly of their policies, which again is fully unrealistic in a context of competition.

- *Potential-field-Guided Deep Deterministic Policy Gradient* (PGDDPG) from Zhou et al. 2021: this algorithm might be the most credible alternative to normal DDPG, as it is designed in two versions, the second featuring no communication between agents. It uses the second critic estimator with Artificial Potential field, a function that gives higher reward to agents that are close to the objectives, making them attracting. If this idea seems quite promising, this algorithm imposes the creation of a totally new theoretical description of the environment in the case of economic competition. Yet, this task seems challenging: how to define attractive points in the environment in situations where firms that are using the algorithms do not know in advance the best price or quantity, as they do not have enough information on the demand or on other firms' production technology? We leave these developments for future work.

Despite the theoretical limitations of normal DDPG, several remarks must be made. First, several studies have, as we will, tried to use the DDPG algorithm in a multi-agent setting. Despite showing that DDPG suffers from high variance and sometimes suboptimal results, Lowe et al. 2017 show that DDPG can still fulfill the task in most cases, using more time, or accepting a second-best equilibrium. If the effects of the improper use of the Deterministic gradient theorem (2.2.4) are not easily measurable, Lowe et al. 2017 analytically proved that in very simplified games, the difference between the true gradient and the true gradients is $P(<\hat{\nabla}J, \nabla J> \geqslant 0) \propto 0.5^N$, which appears tolerable to us. Secondly, in the economic field Graf et al. 2023 has successfully tried to implement a DDPG algorithm in the context of a Bertrand oligopoly, and shown that convergence was archived in most cases, after correctly tuning hyper-parameters. We will try to reproduce and continue this first -to the best of our knowledge-attempt in more complex oligopolies.

3. Objectives and Methodology

3.1. ***Main and secondary objectives***

The main objective of this study is to demonstrate that the (independent) DDPG algorithm can be used to model multi-agent economic markets. Nonetheless, this work is not solely a pure theoretical work. We aim to challenge the algorithm by using it as if it were used by independent firms to automatically determine their prices. Hence, for each market composition (Cournot / Stackelberg), we will study the quality of the convergence of the DDPG algorithm toward an equilibrium from a mathematical point of view, and we will economically question the consequences of such an equilibrium both for the market (with regard to collusion and market concentration) and for the firm itself.

The novelty of this approach is that it is truly *model-free*. We never implement in our simulations any hypothesis on the behavior of agents (rationality and best response for instance), or never give any information on agents (no more than the environment feedback). For instance, in our Cournot model, agents do not even know how many of them are present on the market (and only observe the total quantity produced), or in our Stackelberg model, no information is given on the firms about their status (leader/follower). Hence, by imposing such challenging constraints, this study wants to test both the implementation of learning agents in traditional models, but also the impact of such incomplete and imperfect information on the equilibrium.

To sum up, **we introduce three research questions** that, to the best of our knowledge, lack full answers in the literature:

Q1 Is the DDPG Algorithm a relevant tool in a multi-agent setup, in particular in the case of oligopolies with competition in quantities?

Q2 How does the equilibrium evolve when non-myopic agents are introduced?

Q3 How do learning agents behave in a setting without any stable analytical solution?

3.2. ***Simulation methodology***

To tackle these three research questions, we conduct several batches of simulations (using D, $D()$ and c as demand and cost parameters defined in Section 2.1, and γ the actualization rate from Section 2.2.3):

- First, to challenge the relevance of the DDPG algorithm and answer **Q1** we conduct two types of simulations: a Cournot duopoly, and a Stackelberg duopoly. Both of them should be conducted with $\gamma = 0$ even if it can jeopardize the efficiency of the DDPG algorithm to stick to the game theoretic definition of these games (fully myopic agents).
- Second, we introduce non-myopic agents ($\gamma > 0$) in the Cournot duopoly to evaluate the effect on the equilibrium that we have been previously studied (**Q2**).
- Last, we will focus on the Cournot 4-oligopoly: we will try to simulate chaotic cases ($c \leqslant 0.5$) (**Q3**), and non-myopic agents (**Q2**) to evaluate the effects of these alterations on the equilibrium.

To be very precise, each of our simulations follows the same pattern. The market is only made of firms that are learning agents. If they observe a past value (it is the case in the Cournot oligopoly for instance), it is initialized to 0 at the beginning. All firms are asked to decide their quantities (we call this a "round") and, at the end of the round (i.e. when all firms have decided), they receive from the environment the profit they made and can observe other's quantities. The rounds, when agents are not myopic, are supposed to be all dependent (previous choices can affect next choices) and no run is considered as terminal (in game theoretic terms, we assume that the game is in infinite horizon). Every 128 rounds, a learning session is launched. The simulation reaches its end due to a round limit. There is no convergence test to stop the environment.

Nonetheless, we will in the results section provide some convergence measures. From now on, we will consider as the "convergence value" the mean of the last values taken by the variable when its standard deviation is less than .001. In most cases, this "convergence value" is determined by taking the average of the last 2% values (without taking into account decisions that have been affected by our exploration white-noise process). Contrary to some other approaches, we do not check convergence in simulation time, and hence do not stop the simulation when convergence is reached. All of our simulations are conducted with floating point numbers, with a precision of 10^{-7}.

3.3. ***Hyper-parameters tuning***

Due to our design choices for Neural Networks (we use two fully-connected ReLU layers, following the work of Graf et al. 2023, and one sigmoid output layer), we are constrained to keep our expected outputs in the interval [0,1]. As neural networks will have to learn the Q-Values and the best response function, both Q-Values and possible actions need to stay in this interval in order for the neural networks to be able to approximate them. We could use advanced technics of normalization, but we stick with a simple solution to this problem. First, we use the fact that *argmax* is translation invariant and use an affine transformation to keep rewards in [0,1] while not impacting the optimal behavior to learn (we calibrate η_π and H_π in eq. 3.3.1). Second, we calibrate D and c to find values for $\Pi(q_t^i, q_t^{-i}) \in [0,1]$. This has proven to be a more challenging task that one could have expected, as the feasible set for this constraint is small, especially when the number of agents grows. For future works, we would recommend using a pre-normalization technic to loosen this constraint on possible parameters. As all these values depend on the number of agents and the profit function, they will need to be computed for each different simulation setting, and will hence be provided whenever a batch of simulation is studied.

$$r_t^i(s_t, a_1, ..., a_N) = \eta_\pi \Pi_t^i \left(a_t^i, \sum_{i \neq j} a_t^j \right) + H_\pi \tag{3.3.1}$$

Still on neural networks, we initialize all weights and biases randomly with a normal law. Moreover, our gradient descents and ascents are implemented with a decreasing learning rate (which itself is different for the actor (λ_μ) and for the critic (λ_Q) network), with a decay factor ω^t. These values have been find by a trial-error method, but do not change any valuable results that we manage to get when they are taken in the commonly-accepted interval [.001, .3]. Gradient ascent and descent are implemented with the mini-batch method.

The exploration process appears very determinant in our simulations and might be the most controversial. We enforced a time-dependent process with high variance ($[\sigma(t)]^2$) at the beginning of the simulation, and low variance at the end. Such a mechanism helps the algorithm to converge but can be criticized on two bases: first, it can increase the risk of falling into a local optimum, and then it is not very realistic as it is difficult to imagine a firm accepting errors so big (yet, we accept this hypothesis as firms can train the model in a sandbox with previous data).

We provide a summary of hyper-parameters that have been introduced so far in the Theoretical framework (Table 1). All parameters that have not been elucidated have been found by a trial error method, without any observed influence on any result previously or later mentioned. The influence of the memory buffer maximum size, extensively discussed by Graf et al. 2023, will be studied in the result section.

4. Results

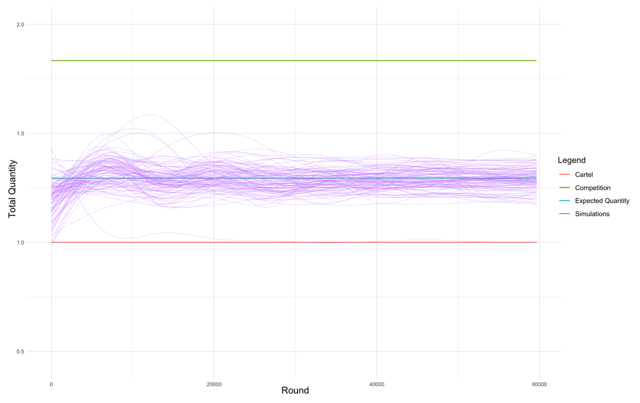
4.1. ***Performance of our DDPG algorithm in standard games with myopic agents***

4.1.1. *The Cournot duopoly*

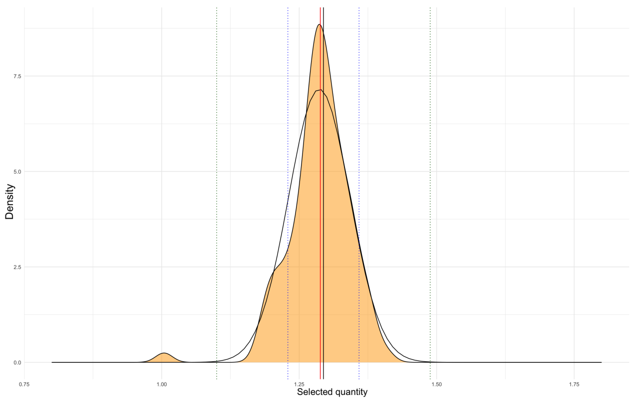
The Cournot duopoly appears to be the simplest and the most obvious oligopoly to model and on which to test our learning agents. Indeed, in the duopoly case, we have a stable analytical solution and a unique equilibrium. As agents are equal in the Cournot oligopoly, our two oligopolists are implemented in mirror, and both directly observe the choice of the other at time $t-1$. As previously explained, to stick to the game theoretic definition of the model, we keep $\gamma = 0$ (fully myopic agents). We use as parameters $D = 2.2$, $c = 0.2$ and compute with eq. 2.1.7, $q_C^* = .647, Q_C^* = 1.294$.

Parameter	Type	Value	Note
D	$\mathbb{R}_+$	-	Demand constant
c	$\mathbb{R}_+$	-	Quad. cost coefficient
$\mathcal{A}$	$\mathbb{R}_+$	$[0, 1]$	Action set
γ	$[0, 1]$	-	Future discount factor
$m = \text{card}(\mathcal{B})$	$\mathbb{N}$	-	Memory buffer max. size
T	$\mathbb{N}$	$[\![6 \times 10^4, 12 \times 10^4]\!]$	Time limit
ς	$[0, 1]$	$\{.06, .15, .3\}$	Exploration noise max. variance
$\sigma(t)$	$(\mathbb{R}_+)^{\mathbb{N}}$	-	Exploration noise variance
η_π	$\mathbb{R}$	[0,1]	Profit reward scaling factor
H_π	$\mathbb{R}$	[0,1]	Profit reward scaling factor
θ_π	$\mathbb{R}_+^x$	$\mathcal{N}(0, 0.3)$	Actor's N. N. initialization weights
θ_μ	$\mathbb{R}_+^x$	$\mathcal{N}(0, 0.3)$	Critic's N. N. initialization weights
τ	$[0, 1]$	.95	Polyak averaging coefficient
λ_μ	$[.001, .3]$	.001	Actor network learning rate
λ_Q	$[.001, .3]$	.1	Critic network learning rate
ω	$[0, 1]$	.9998	Learning rate decay factor
$\text{card}(B)$	$\mathbb{N}$	32	Batch size
β	$\mathbb{N}$	32	Batches per training episode

TABLE 1. **Hyper-parameters summary.**
All parameters with values "-" are implementation specific, i.e. will be adapted depending on the model studied.



(A) **Evolution of the total quantity $(q_t^1 + q_t^2)$.** The Cournot predicted quantity is in blue, the cartel equilibrium in red and the perfect competition equilibrium in green.



(B) **Distribution of the total quantity $(q_t^1+q_t^2)$ at convergence.**
In black a normal law with the distribution's parameters (μ, σ) and the Cournot expected quantity; in red the median of simulations; in blue the 5% confidence interval; in green the 15%; in orange the density function of $(q_t^1 + q_t^2)$.

FIGURE 4.1.1. Quantity chosen by the learning agents in a Cournot duopoly (107 simulations conducted with $m = 2000$, $\varsigma = .15$, $\gamma = 0$)

Overall, our simulations reach an equilibrium that is in a neighborhood of the analytical equilibrium value (fig. 4.1.1). The results are more accurate when studied at the level of the market and not at the firm's level. Indeed, the equilibrium reached is never perfectly symmetrical, and during training, one firm always ends up with a small advantage. We provide some distribution measures in Table 2, with a Shapiro-Wilk normality test (Shapiro and Wilk 1965) to check whether it is reasonable to consider that our simulations converge toward an equilibrium following a normal law.

	Global dispersion			Shapiro test		Distribution	
Sample	$Q_C^* \pm 5\%$	$Q_C^* \pm 10\%$	$Q_C^* \pm 15\%$	W	p-value	$\bar{x}$	$\hat{s}$
Uncorrected	0.794	0.991	0.991	0.932	3.871e-05	1.287	0.0557
Without outlier	0.802	1	1	0.990	0.613	1.289	0.0487

Table 2. **Descriptive statistics of the simulation sample of the Cournot duopoly** (107 simulations conducted with $m = 2000$, $\varsigma = .15$, $\gamma = 0$).
79.4% of simulations have converged toward an equilibrium that is in the interval $Q_C^* \pm 5\%$ with Q_C^* the Cournot analytical total solution.

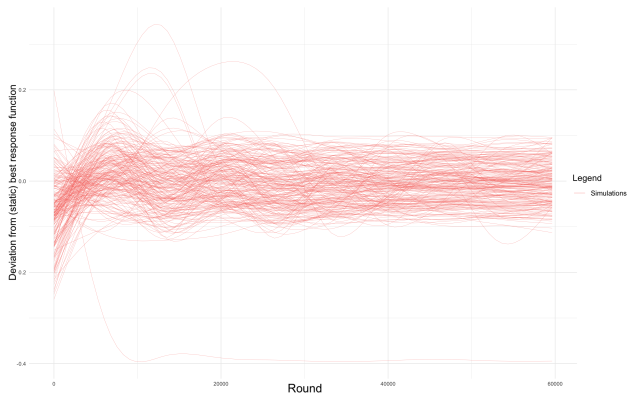
As it can be seen in our density plot 4.1.1, there is one outlier, one simulation that has converged toward an equilibrium that is clearly lower than what we should expect. In fact, the simulation has converged toward the Cartel equilibrium. This result is quite surprising as agents are myopic and do not cooperate, but can be explained by a low exploration that can have biased the training (has agents are trained synchonously). When excluding this simulation, we can no longer reject the hypothesis that our simulations converge following a normal law around the total value of the Cournot model (according to the Shapiro test in table 2). If we consider these simulations as an estimator for the Nash equilibrium, it appears that it is an asymptotic unbiased estimator ($Q_C^* - \bar{x} = .005$) and a consistent estimator.

In game theoretic terms, the expected consequence of these results should be that agents are learning their analytical best response functions and behave accordingly. Nonetheless, a closer look into the policies that has been learned by agents show that the DDPG algorithm is quite efficient to estimate locally the best action, but poorly behaves for extreme values (fig. 4.1.2). The further we get from the simulation equilibrium, the worst becomes the estimation of the optimal policy.

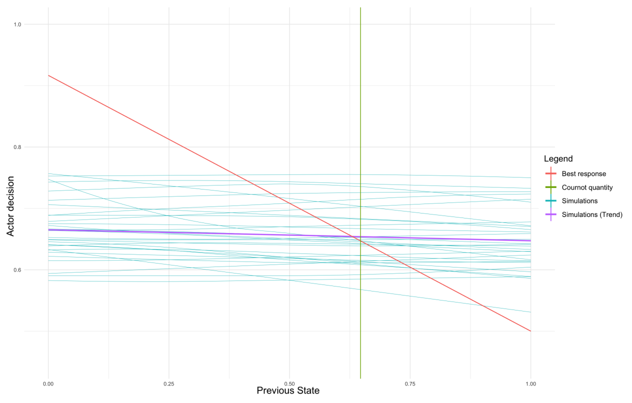
The same issue arises in the Critic estimator (fig. 4.1.5). It seems that the DDPG algorithm manages quite well to predict low profit values and near-equilibrium values, but still struggle with extreme profits. This is quite expected as rounds featuring a very high level of profit (that is higher than the profit in the Cournot equilibrium) must have been quite rare in the learning stage, as they need the other agent to suffer losses for a long time, which would be irrational. These estimation errors are the symptom of the biggest issue with our technic: as learning is synchronous, agents can only learn rewards when other agents are leading them to it. Hence, some equilibria can remain undiscovered if other agents never play what it is needed to find them.

Hyper-parameters influence As we have shown that the DDPG algorithm was in fact a great tool to model the Cournot duopoly because learning agents are behaving quite as predicted by the analytical results (they reach the Nash equilibrium demonstrating rationality and good understanding of the problem), we now want to emphasize the effect of some parameters on the behavior of the algorithm, and try to improve learning performance outside of the neighboring of the equilibrium.

First, one important parameter is the exploration policy. To learn the optimal policy, agents need to explore the action space. Several policies have been implemented in the literature. Here,



(A) **Deviation from static best response over time.**
Computed using eq. 2.1.6.



(B) **Optimal policy at convergence.**
In red is the analytical best response (2.1.6), in blue are the simulations' output. We can see that the further we get from the equilibrium, the worst is the estimation.

FIGURE 4.1.2. **Optimal policy learned by agents in a Cournot duopoly** (107 simulations conducted with $m = 2000$, $\varsigma = .15$, $\gamma = 0$).
The left plot shows that, as much as time passes, agents are selecting actions that are closer and closer to the best response values ($\pm .1$). Nonetheless, the right plot reminds that DDPG is not very efficient to estimate the best response function when we look to values that are far from the equilibrium point (the distance between blue lines and the red line is increasing).

as described in section 1.2 as the parameter $\mathcal{G}$, we use a normally distributed white noise to force the agents to explore the action space and not to stick to their greedy value. The variance of the white noise is determined by $\sigma^{\text{LIN}}(t) = \varsigma(1 - \frac{t}{T})$ and is decreasing as time passes. We test several values for ς, which is the maximum standard deviation of the white noise of our model. To assess the performance of the algorithm in each case, we focus on the same criteria as before, but implement a new measure that tries to take into account the difference between the actions of the two players. Indeed, in our simulations, if the total quantity has well "converged" toward what the model predicted, the individual quantites are almost never equal, and we want to take into account this distance between the two selected quantity at the end of the simulation. We define eq. 4.1.1 which computes a score by selecting the two agents that choose on average the most distant quantities in each simulation $s \in \mathcal{S}$ and by taking the average of these values. Results are given in Table 3 and show that $\varsigma = .15$ is the optimal value to minimize dispersion but that $\varsigma = .3$ was actually quite close in terms of performance. Nonetheless, we will keep as a rule of thumb that $\varsigma = .15$ is an optimal value for simple problems in terms of dispersion but can lead to some under-exploration issues as already seen in the critic analysis. For more complex models, we will keep $\varsigma = .3$ as the extra-exploration comes with a very little cost in terms of dispersion.

$$\breve{S} = \frac{1}{\text{card}(\mathcal{S})} \sum_{s \in \mathcal{S}} \max_{i,j \in \mathcal{A}} \left(\frac{1}{T + 1 - .98 \cdot T} \sum_{t \in [\![.98 \cdot T, T]\!]} ||q_t^i - q_t^j|| \right) \tag{4.1.1}$$

In Table 3, we also provide the results of the implementation of a non-linear variance decay $\sigma^{\text{EXP TRUNC}}(t) = \varsigma \max\left\{1 - 1.0001^{t - \frac{2}{3}T}; 0\right\}$. This function maintains for longer a higher variance, and features a drastic fall at two thirds of the simulation. This non-linear decay is

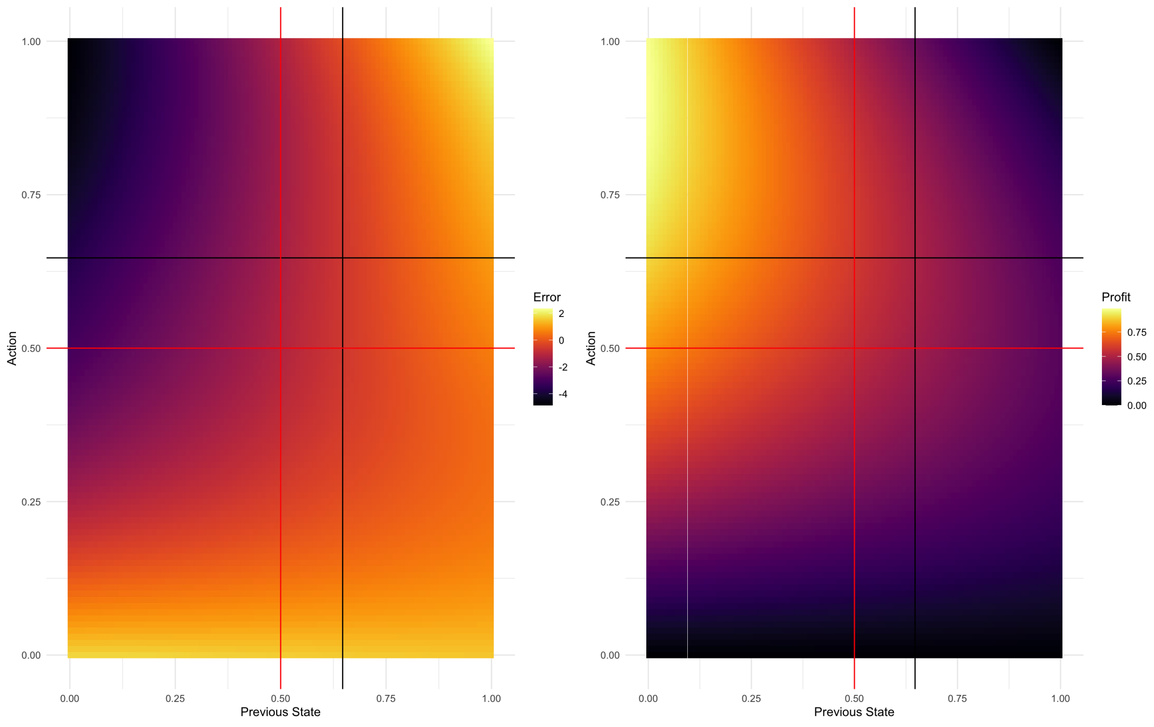


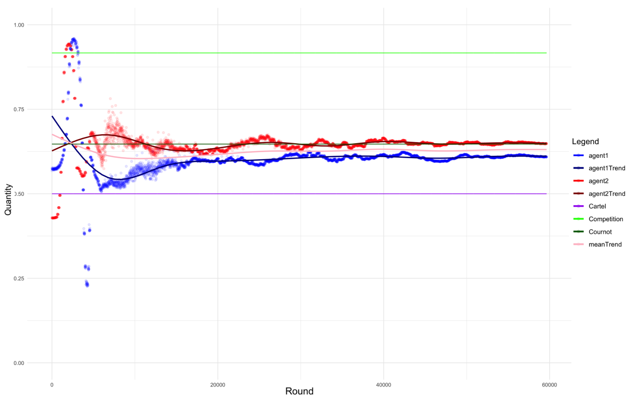
FIGURE 4.1.3. **Estimated critic of agents in a Cournot duopoly** (subsample of 15 critics over 107 simulations conducted with $m = 2000$, $\varsigma = .15$, $\gamma = 0$). Black lines are the Cournot equilibrium, red lines are the Cartel equilibrium. On the left, the heat-map represents the average estimation error (standardized), whereas the right heat-map reminds profit values (the previous state being the previous quantity selected by the opponent).

interesting for our analysis for two main reasons: first, it is economically realistic (firms will want to learn rapidly to stop using random action as fast as possible to avoid losses), and it allows the algorithm to have a better and longer exploration period while not enlarging the simulation duration. This exploration policy is interesting as it features a lower dispersion but is less precise. We can state that this policy is not optimal in the setting of fully myopic agents in simple settings as a Cournot duopoly.

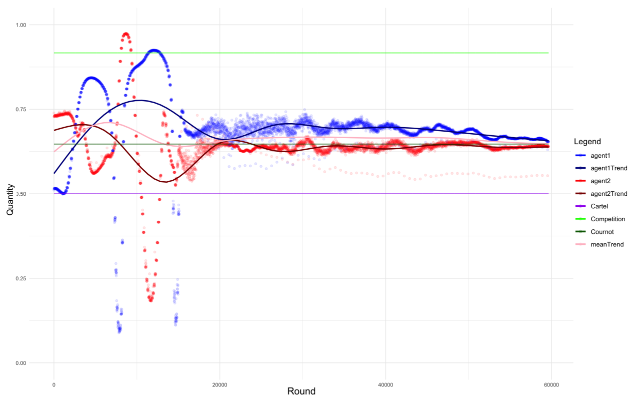
	Global dispersion			Intra-dispersion score			
Sample	$Q_C^* \pm 5\%$	$Q_C^* \pm 10\%$	$Q_C^* \pm 15\%$	Q1($\breve{S}$)	median($\breve{S}$)	Q3($\breve{S}$)	$\#\mathcal{S}$
Lin. $\varsigma = .06$	0.583	0.667	0.667	0.031	0.056	0.223	12
Lin. $\varsigma = .15$	0.794	0.991	0.991	0.029	0.063	0.126	107
Lin. $\varsigma = .3$	0.75	1	1	0.043	0.064	0.135	10
Trunc. Exp. $\varsigma = .3$	0.58	0.75	1	0.041	0.058	0.130	12

TABLE 3. **Influence of the parameter ς and of the choice of the white noise decay method on the Cournot duopoly** ($m = 2000$, $\gamma = 0$). 79.4% of simulations have converged toward an equilibrium that is in the interval $Q_C^* \pm 5\%$ with Q_C^* the Cournot analytical total solution in the $\varsigma = .15$ scenario. Q1 and Q3 are the first and the third distribution quartiles.

The other hyperparameter that have been already studied in the literature (especially by Graf et al. 2023) is the memory buffer size. To truly stick to the requirements of the game theoretic definition of the games that we study, we should not implement memory at all. But, as DDPG algorithm rely on neural networks to learn optimal policy, it is technically impossible to not use memory (Lillicrap et al. 2015) for training the algorithm. Graf et al. 2023 have stated that the memory buffer size should only affect convergence time and not convergence properties or accuracy. In our simulations (fig. 4.1.4), we find same results, that tends to indicate that this is a feature of the DDPG algorithm and not implementation specific. As we do not bring new results here, we do not perform an in-depth review of the impact of buffer size on our algorithm and encourage the interested reader to study the previous reference.



(A) $m = 500$



(B) $m = 2000$

FIGURE 4.1.4. The impact of memory buffer size in a Cournot duopoly (single simulation conducted with $\varsigma = .15$, $\gamma = 0$).
Lecture: As stated by Graf et al. 2023, we observe that the stabilization around the "convergence value" is reached faster when the memory buffer is smaller.

4.1.2. *The Stackelberg duopoly*

We now try to use our DDPG Algorithm in the Stackelberg framework. This model is more challenging than the Cournot one as one agent has to play first. There exists, to the best of our knowledge, no consensus nor example of implementation of continuous actor-critic algorithm in the Stackelberg framework. As we want our model to stick as much as possible with the game theoretic definition of the game, we choose the following design with fully myopic agents: the *leader* chooses its quantity first with a constant input (and not the previous quantity as we did in the Cournot model) and the *follower* reacts by choosing its quantity with the other's quantity as input. The following design can be written with the following system of simple rules:

$$\forall t \in T, \begin{cases} s_t^L = 0 \\ s_t^F = a_t^L \end{cases} \tag{4.1.2}$$

The results of our simulations are to some extent disappointing. This model is quite a bad match with the DDPG algorithm in synchronous learning (and with a model-free implementation). Indeed, if the *follower* tends to learn quite well the best response as they do in the Cournot framework, performances are far worse for the leader that fails more often and seems to exhibit a bias. Such results are not entirely surprising as the algorithm is not well designed to be used with rules that are as simple as are eq. 4.1.2. The DDPG algorithm appears not well suited to deal with single-state problems. We provide a summary of our findings in Table 4. The Shapiro tests confirm this diagnostic: if the follower converges following a normal law, the leader seems more unstable. The negative bias observed is due to the leader, the follower playing the best response to the leader's action. Overall, the total quantity seems pretty well approximated, despite the issues with the leader.

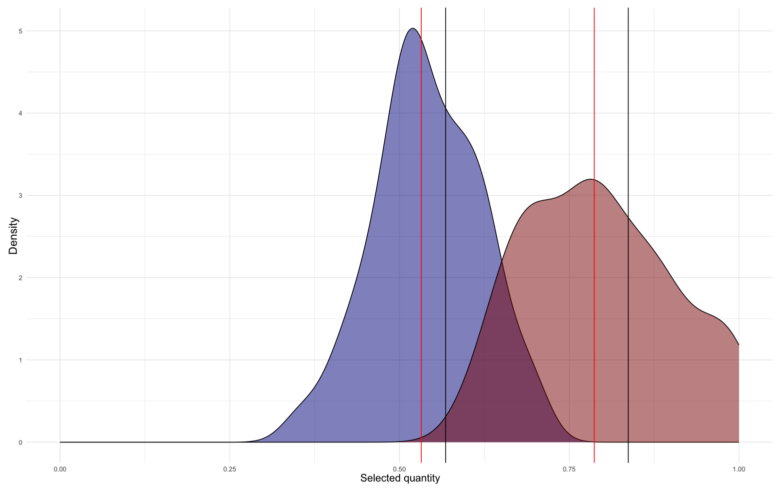


FIGURE 4.1.5. **Distribution of selected quantities in the Stackelberg duopoly** (with the follower in blue and the leader in red, 107 simulations conducted with $m = 500$, $\varsigma = .3$, $\gamma = 0$)
Black lines are the analytical optimal quantities, red lines are the median of observed distributions.

		Dispersion			Shapiro Test		Distribution	
Sample	q_S^*	$q_S^* \pm 5\%$	$q_S^* \pm 10\%$	$q_S^* \pm 15\%$	W	p-value	$\bar{x}$	$\hat{s}$
Follower	0.568	0.214	0.536	0.726	0.9891	0.7083	0.541	0.077
Leader	0.837	0.202	0.512	0.619	0.961	0.013	0.795	0.107
Total quantity	1.405	0.536	0.821	0.940	0.978	0.159	1.337	0.083

TABLE 4. **Implementation performance of the DDPG Algorithm in the Stackelberg duopoly** (84 simulations, $m = 500$, $\gamma = 0$, $\varsigma = .3$).
72.6% of simulations have converged toward an equilibrium where the follower has chosen its quantity in the interval $q_S^* \pm 5\%$ with q_S^* the Stackelberg analytical solution.

4.2. *Non-myopic agents in Cournot games: a study of algorithmic collusion*

Now that the DDPG algorithm have been proven relevant in the modeling of oligopolies in the myopic-agents framework, we will try to assess the evolution of the behavior of the algorithm and of the outcome (equilibrium) with agents that are gradually less and less myopic. The aim of this part is to answer our second research question (**Q2**).

First, we need to introduce a slight variation from the theoretical background that we have described in previous sections. In order to keep our inputs and expected outputs in our $[0, 1]$ interval, we alter the Bellman equation such that eq. 2.2.1 becomes:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}\sim E}[(1-\gamma)r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, \mu(s_{t+1}))|s_t, a_t] \tag{4.2.1}$$

This alteration has been proven almost effectless on the output in our simulations. Nonetheless, it allows us to avoid advanced technics to scale the critic output. Moreover, it does not jeopardize the theoretical convergence of the Bellman equation as it keeps γ lower than 1, insuring that the operator stays a contraction mapping. To keep our results comparable with other works on collusive behavior with RL-algorithms, we can consider as a rule of thumb that our case $\gamma = .5$ is more or less equivalent with the normal case of $\gamma = .99$[2].

Our objective is here double: we want to assess the impact of γ on the ability of the DDPG algorithm to find an equilibrium, and we want to see how the equilibrium reached is altered. To assess these two points, we use two groups of measures. First, we provide for each group of simulation measures of intra-dispersion (using 4.1.1) to assess whether the selected quantities were different of each other, and inter-dispersion ($Q^*_{\text{Co.}} \pm k\%$, and the p-value of the Shapiro test) to assess how and whether the simulations converge toward a fixed equilibrium. Second, we assess the deviation of the equilibrium by: first, the share of simulations that reach an equilibrium that is near the Cartel equilibrium, and second, the Δ-score from Calvano, Calzolari, and Denicolò 2019 which takes into account the deviation from the Cournot equilibrium.

$$\Delta = \frac{\bar{\Pi} - \Pi^*_{\text{Cournot}}}{\Pi^*_{\text{Cartel}} - \Pi^*_{\text{Cournot}}} \tag{4.2.2}$$

Our simulations are conducted with $D = 2.2$ and $c = 0.2$, for which the analytical equilibrium ($q^*_{\text{Comp.}}, \pi^*_{\text{Comp.}}, q^*_{\text{Cournot}}, \pi^*_{\text{Cournot}}, q^*_{\text{Cartel}}, \pi^*_{\text{Cartel}}$), can be computed as (0.917, 0.165, 0.647, 0.492, 0.500, 0.539). The choice of these parameters has been made to ensure that all possible values lay in the interval $[0, 1]$ and, for the 4-oligopoly, that the equilibrium was analytically convergent. All simulation are conducted with $m = 1000$, with an exploration variance decay policy of $\sigma^{\text{EXP TRUNC}}(t) = \varsigma \max\left\{1 - 1.0001^{t-0.85T}; 0\right\}$, with a larger number of rounds $T = 9 \times 10^4$ and, thus, a larger learning rate decay factor of $\omega = .9999$. These parameters have been set to ensure that the algorithm can have sufficient training to reach an equilibrium.

We provide in Table 5 all the descriptive statistics previously defined. The first result is perhaps the most expected: when γ goes beyond the .5 threshold, the consistency of the algorithm is jeopardized, as the target network has more weight than the normal critic network, which leads to an expected instability that can be spotted by our intra-dispersion measure and the higher variance in the equilibria obtained. We also spot strange results for the case $\gamma = .2$ which seems discontinuous with the other values without us being able to provide any logical explanation.

On the side of the collusion analysis, and more generally of the alteration of the equilibrium, we observe that all simulations with "consistent" values of γ (according to the last paragraph, $\gamma < .6$) have converged toward an equilibrium that is slightly lower than what the model would expect (the Cournot total value) and then what we encountered in the last section (almost every group has converged to a mean lower than 1.28). This first observation is corroborated by the Δ-score (which tries to measure the deviation from the Nash equilibrium to the Cartel equilibrium) that is on average above .20 except for the strange $\gamma = .2$ case. When we conduct a Welch t-test, we find that this difference is statistically significative in almost every group. Hence, we can state that our simulations have reached a collusive equilibrium.

Nonetheless, Table 5 is designed to allow us to conduct a counterfactual analysis, with a perfect counterfactual (the case $\gamma = 0$). This counterfactual is designed to allow us to test

[2]In some previous works including the one of Calvano, Calzolari, and Denicolò 2019, the γ parameter can be denoted as δ.

Sample			Inter-dispersion				Distribution		Intra-dispersion		
			$Q^*_{\text{Co.}}$	$Q^*_{\text{Co.}}$	$Q^*_{\text{Co.}}$	Shapiro					
γ	m	T	$\pm5\%$	$\pm10\%$	$\pm15\%$	p-value	$\bar{Q}$	sd(Q)	mean($\breve{S}$)	sd($\breve{S}$)	$\#\mathcal{S}$
0	1000	9×10^4	0.55	0.85	0.95	0.053	1.273	0.092	0.121	0.07	74
0.1	1000	9×10^4	0.64	0.95	0.97	0.122	1.274	0.076	0.151	0.089	39
0.2	1000	9×10^4	0.54	0.87	0.93	0.44	1.295	0.092	0.156	0.08	54
0.3	1000	9×10^4	0.64	0.92	1	0.4	1.27	0.063	0.152	0.101	36
0.4	1000	9×10^4	0.6	0.96	1	0.556	1.281	0.072	0.15	0.081	48
0.5	1000	9×10^4	0.47	0.86	0.93	0.78	1.266	0.094	0.15	0.089	58
0.6	1000	9×10^4	0.4	0.73	0.97	0.492	1.267	0.103	0.169	0.12	30
0.7	1000	9×10^4	0.49	0.76	0.95	0.986	1.305	0.105	0.173	0.15	55
0.8	1000	9×10^4	0.53	0.82	0.97	0.219	1.284	0.09	0.2	0.13	34
0.5	2000	9×10^4	0.57	0.79	0.97	0.205	1.294	0.095	0.15	0.093	72
0.5	3000	9×10^4	0.59	0.97	1	0.622	1.294	0.069	0.156	0.118	29
0	500	6×10^4	0.69	0.94	0.99	0	1.313	0.083	0.136	0.096	72
0	1000	6×10^4	0.64	0.91	0.97	0.015	1.295	0.081	0.149	0.098	150

(A) Implementation performance

Sample			Inter-dispersion				Collusion		t-test p-value		
			$Q^*_{\text{Car.}}$	$Q^*_{\text{Car.}}$	$Q^*_{\text{Car.}}$						
γ	m	T	$\pm5\%$	$\pm10\%$	$\pm15\%$	$\bar{\Pi}$	$\bar{\delta}$	$\bar{\Delta}$	$\bar{\Delta} > \bar{\Delta}_{\gamma=0}$	$\bar{\Delta} > 0$	$\#\mathcal{S}$
0	1000	9×10^4	0.01	0.05	0.12	1.005	0.939	0.214	-	0.001	74
0.1	1000	9×10^4	0.03	0.03	0.03	1.008	0.946	0.252	0.356	0.003	39
0.2	1000	9×10^4	0	0.04	0.09	0.992	0.96	0.074	0.903	0.399	54
0.3	1000	9×10^4	0	0	0	1.012	0.947	0.292	0.207	0	36
0.4	1000	9×10^4	0	0	0.02	1.004	0.965	0.212	0.511	0.005	48
0.5	1000	9×10^4	0.03	0.07	0.09	1.009	0.973	0.261	0.322	0.001	58
0.6	1000	9×10^4	0	0	0.13	1.006	0.979	0.225	0.471	0.104	30
0.7	1000	9×10^4	0.02	0.02	0.05	0.981	0.985	-0.042	0.978	0.696	55
0.8	1000	9×10^4	0	0.03	0.09	0.997	0.995	0.13	0.758	0.211	34
0.5	2000	9×10^4	0	0	0.11	0.992	0.967	0.077	0.904	0.357	72
0.5	3000	9×10^4	0	0	0	0.996	0.978	0.122	0.794	0.195	29
0	500	6×10^4	0	0	0	0.982	0.952	-0.028	0.984	0.763	72
0	1000	6×10^4	0.01	0.02	0.03	0.994	0.953	0.099	0.928	0.034	150

(B) Collusion assessment

TABLE 5. **Counterfactual analysis of the effects of non-myopic agents with our DDPG Algorithm** ($\varsigma = .3$, 751 simulations).
The two last lines are implemented as a reminder, to assess the effect of the increase of the simulation length.

whether there exists a consistent relationship between γ and collusive equilibrium (assessed here by the Δ-score), as proved with Q-Learning by Calvano, Calzolari, and Denicolò 2019. Here, we do not find any evidence of such a relationship. To test this result, we conduct asymmetric

Welch t-tests with reference to our counterfactual $\gamma = 0$ (we denote this test in Table 5b as "$\bar{\Delta} > \bar{\Delta}_{\gamma=0}$") and find no statistically significant difference (fig 4.2.1). We also try to use an OLS estimation using a simple model $\Delta_i = \beta_0 + \beta_1\gamma + \epsilon$ and find a non-significative estimate of $\hat{\beta}_1 = 0.033$ (st. error of .15 and p-value of .82 on $\gamma \in [0, 7)$ and $\gamma \neq .2$). Our results are here more in line with the ones from Abada, Lambin, and Tchakarov 2022 who have not found any relationship between γ and collusive equilibrium with an actor-critic implementation.

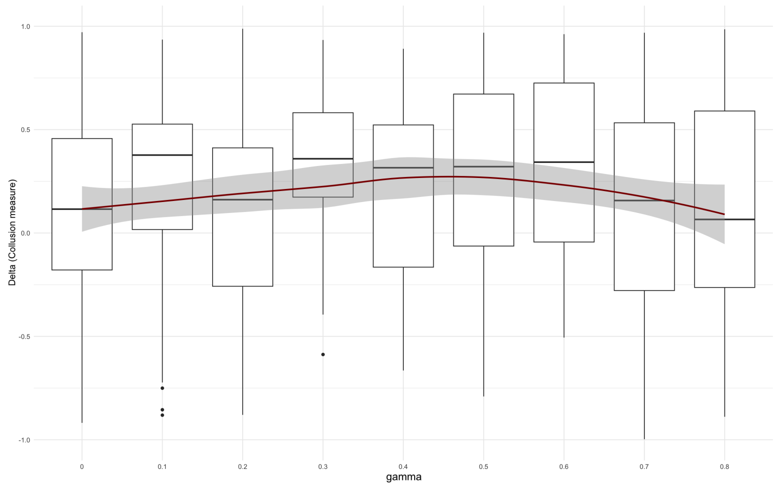


FIGURE 4.2.1. **The relation between the γ parameter and the Δ-score** (518 simulations conducted with $m = 1000$, $\varsigma = .3$)

Though we cannot find any proof of a link between γ and the collusive outcome, we can state that the increase of the simulation duration (and thus of the training) and of the memory buffer size has caused the statistically significant increase of the Δ-score. We use the counterfactuals $\{m = 1000, \gamma = 0, T = 6 \times 10^4\}$ and $\{m = 500, \gamma = 0, T = 6 \times 10^4\}$ in Table 5b, and show that their Δ-score is significantly lower than the one of the group $\{m = 1000, \gamma = 0, T = 9 \times 10^4\}$. The increase of the p-value of the Shapiro test is explained by the fact that there exist two attractive points in some of our simulations: a Nash equilibrium where most simulations are converging to, and sometimes (though it is rare) the Cartel equilibrium. Removing these extreme values corrects the issue.

We also perform simulations with $m = 2000$ and $m = 3000$ to test whether it is relevant to keep increasing the memory length to increase collusion. We find no evidence of such a relationship. Worse, we find that collusion was actually smaller in these groups than in the $m = 1000$ group. We find that the average Δ-score was significantly lower in the $m = 2000$ group than it is in the $m = 1000$ (Welch's test p-value of 0.054). We also find that we cannot conclude that the average total profit collected by firms is higher than what it should analytically be at the Nash equilibrium (we find a p-value of .155 and .08 against "sample average profit is greater than .984" in a Welch t-test). This suggests that collusion is only possible with rare tuples of values for (m, T, ω) and is in fact quite rare.

Despite having no significant effect on the outcome, the γ parameter is not entirely useless. Indeed, we tried to assess the evolution of the Critic estimator when γ increased (with perfect counterfactuals as every other parameter is kept constant). We implemented a new measure, δ that takes into account how the algorithm is valorizing the collusive outcome against the Nash equilibrium. Our δ is simply the quotient of the average of the estimated Q-Value in an ϵ-neighborhood around the Cartel quantity divided by the same average in the ϵ-neighborhood of the Nash quantity (eq. 4.2.3). We take for simplicity the neighborhood $\pm 15\%$.

$$\delta = \frac{\sum_{s_t \in B(q^*_{Ca.}, 0.15 q^*_{Ca.})} \sum_{a_t \in B(q^*_{Ca.}, 0.15 q^*_{Ca.})} Q^\pi(s_t, a_t)}{\sum_{s_t \in B(q^*_{Co.}, 0.15 q^*_{Co.})} \sum_{a_t \in B(q^*_{Co.}, 0.15 q^*_{Co.})} Q^\pi(s_t, a_t)} \tag{4.2.3}$$

We find a statistically significant relationship between δ[3] and γ. When performing a simple OLS regression $\delta_i = \beta_0 + \beta_1\gamma + \epsilon_i$, on the $\{m = 1000, \gamma \in [0, 0.7], T = 9 \times 10^4\}$ dataset, we estimate $\hat{\delta_i} = 0.938 + 0.066\gamma_i + e_i$ with a high significance for both regressors, especially for $\hat{\beta}_1$ whose p-value is less that $2e^{-16}$ and its standard error is 0.006. This property appears to be also true even in groups without any trace of collusion like the ones with $m = 2000$ and $m = 3000$. This suggests that the algorithm is better valorizing the collusive equilibrium as γ increases, but despite being more and more valuable, chooses not to play it.

To delve more into the details of this relationship, we implement two linear models:

$$\min(\delta_i, \delta_{-i}) = \beta_0^f + \beta_1^f \max(\delta_i, \delta_{-i}) + \beta_2^f \gamma + \epsilon^f \tag{4.2.4}$$

$$a_i = \beta_0^S + \beta_1^S \delta_i + \beta_2^S \delta_{-i} + \epsilon^S \tag{4.2.5}$$

The first one, eq. 4.2.4 links the minimum of (δ_i, δ_{-i}) with the maximum value of (δ_i, δ_{-i}), controlling by the parameter γ of the simulation. This tries to encompass the relationship between the two variables δ_1 and δ_2: we want to check whether they covariate in the same direction, i.e. whether both algorithms are together better valorizing cooperation. The second model, eq. 4.2.5, wants to show that the action of each agent is both affected by its own valuation of the cooperative equilibrium and the valuation of its opponent. These two models allow us to evaluate the global effect of these valuations on the total equilibrium.

If we assume without any loss of generality that $\delta_i \leqslant \delta_{-i}$:

$$\begin{cases} a_i = \beta_0^S + \beta_1^S \left(\beta_0^f + \beta_1^f \delta_{-i} + \beta_2^f \gamma + \epsilon^f\right) + \beta_2^S \delta_{-i} + \epsilon^S \\ a_{-i} = \beta_0^S + \beta_1^S \delta_{-i} + \beta_2^S \left(\beta_0^f + \beta_1^f \delta_{-i} + \beta_2^f \gamma + \epsilon^f\right) + \epsilon^S \end{cases} \tag{4.2.6}$$

$$\Leftrightarrow \begin{cases} a_i = \beta_0^S + \beta_1^S \beta_0^f + (\beta_1^S \beta_1^f + \beta_2^S)\delta_{-i} + \beta_1^S \beta_2^f \gamma + \beta_1^S \epsilon^f + \epsilon^S \\ a_{-i} = \beta_0^S + \beta_2^S \beta_0^f + (\beta_1^S + \beta_2^S \beta_1^f)\delta_{-i} + \beta_2^S \beta_2^f \gamma + \beta_2^S \epsilon^f + \epsilon^S \end{cases} \tag{4.2.7}$$

$$\begin{aligned} \Rightarrow a_i + a_{-i} = {} & 2\beta_0^S + (\beta_1^S + \beta_2^S)\beta_0^f + (\beta_1^S + \beta_2^S)(1 + \beta_1^f)\delta_{-i} \\ & + \beta_2^f(\beta_1^S + \beta_2^S)\gamma + (\beta_1^S + \beta_2^S)\epsilon^f + 2\epsilon^S \end{aligned} \tag{4.2.8}$$

To estimate these models, we split our sample into two groups. The first one is constructed within the group $\gamma = 0, m = 1000, T = 6 \times 10^4$ presented in Table 5, and which is our "perfect-collusion-counterfactual" as it is a group where the total profit (and action) converges toward the analytical Nash solution, and in which the Δ-score is on average 0 (or slightly above for $m = 1000$, but this increase is dramatically slighter than what is observed in the other groups). It is also the group in which the perfect tuple enabling collusion is not met. We remove from this group any simulation in which the Δ-score reached a value above .3. We denote it hereinafter as the "Zero-collusion group". We construct another group with all the simulations performed with the settings $\gamma \in [0, 0.7], m = 1000, T = 9 \times 10^4$ where we reached average Δ-scores statistically significantly different from 0. Our goal in this estimation is to analyse how being in the optimal tuple $(m = 1000, T = 6 \times 10^4, \omega = .9999)$ affects the decisions of agents and makes them cooperate more.

[3] Here, it is the mean of the δ-score of each of the two agents.

	Estimate	t. value	p-value
(cste)	0.271	2.73	0.0077
	(0.099)		**
$\max(\delta_i, \delta_{-i})$	0.685	6.70	$1.85e^{-9}$
	(0.102)		***
γ	-	-	-
	(-)		

(A) Zero-collusion group (91 simulations, $R_a^2 = .33$)

	Estimate	t. value	p-value
(cste)	0.444	7.61	$2.14e^{-13}$
	(0.058)		***
$\max(\delta_i, \delta_{-i})$	0.486	8.03	$1.17e^{-14}$
	(0.061)		***
γ	0.058	7.66	$1.49e^{-13}$
	(0.008)		***

(B) Collusion group (393 simulations, $R_a^2 = .34$)

TABLE 6. **First stage regression to link** $\min(\delta_i, \delta_{-i})$ **with** $\max(\delta_i, \delta_{-i})$ **(eq. 4.2.4).**

	Estimate	t. value	p-value
(cste)	0.425	2.43	0.016 *
	(0.175)		
δ_i	-1.139	-8.04	$1.22e^{-13}$ ***
	(0.142)		
δ_{-i}	1.398	9.86	$< 2e^{-16}$ ***
	(0.142)		

(A) Zero-collusion group (182 observations, $R_a^2 = .43$)

	Estimate	t. value	p-value
(cste)	0.478	5.35	$1.17e^{-07}$ ***
	(0.089)		
δ_i	-1.107	-15.84	$< 2e^{-16}$ ***
	(0.07)		
δ_{-i}	1.275	18.25	$< 2e^{-16}$ ***
	(0.07)		

(B) Collusion group (788 observations, $R_a^2 = .4$)

TABLE 7. **Second stage regression to link** a_i **with** (δ_i, δ_{-i}) **(eq. 4.2.5).**

We provide estimation results for the two linear models in Table 6 for the first stage, and Table 7 for the second stage. In every estimation, regressors (except the constant) are always very significant with very low p-values. From Table 6, we can state with confidence that δ_i and δ_{-i} are positively correlated. This shows that both algorithms are increasing their valuation of the cooperative state at the same time. This result, if taken without the second stage, could be misleading: even if they are together better valorizing the cooperative state, they do not necessary play it.

Indeed, Table 7 allows us to recover a famous result in game theory: cooperative equilibria can have two opposite effects on the decision of the player. First, they allow it to increase its profit, what incentivizes it to lower its quantity (we find it in our results as β_1^S, the coefficient of δ_i on a_i, is always negative, indicating a reduction of the produced quantity towards the Cartel quantity) to reach the equilibrium. Second, they are an opportunity to increase its profit by taking advantage of the other player. Indeed, if the other will play cooperatively, and if our agent at the same time chooses to produce more, our agent will receive a higher profit, incentivizing him to increase its quantity (we find it in our results as β_2^S, the coefficient of δ_{-i} on a_i, is always positive, indicating an increase of the produced quantity).

These two effects (incentive to cooperate or temptation to betray) are playing simultaneously making the task to determine the outcome difficult. We write the system of our two agents in eq. 4.2.6 and find by using eq. 4.2.8 that the effect of the variables δ_i, δ_{-i} on the total quantity can be identified by the parameter $(\beta_1^S + \beta_2^S)(1 + \beta_1^f)$, whose magnitude determine the effect of an increase in the valorization of the collusive equilibrium. Due to the complex computation that has been required to find such result, this parameter cannot be interpreted by itself, but should be compared between groups. According to Tables 6 and 7, we can estimate this parameter:

$$(\hat{\beta}_1^S + \hat{\beta}_2^S)(1 + \hat{\beta}_1^f) = \begin{cases} (-1.139 + 1.398)(1 + 0.685) = 0.436 & \text{Zero-collusion group} \\ (-1.107 + 1.275)(1 + 0.486) = 0.250 & \text{Collusion group} \end{cases} \tag{4.2.9}$$

By eq. 4.2.9, we find that $a_i + a_{-i}$ will increase more in the group of no collusion when the δ-score increases, proving an higher prevalence of the betray decision when the cooperative outcome is at reach. To be perfectly rigorous, we estimate the constant term (and the "quasi-constant" part of γ as it is insignificant at a precision of 1×10^{-4}). We find in eq. 4.2.10 that the constant part was lower in the Zero-collusion group, jeopardizing the interpretation of the $(\hat{\beta}_1^S + \hat{\beta}_2^S)(1 + \hat{\beta}_1^f)$ coefficient. Nonetheless, we can easily show that this difference is insignificant: by considering that 75% of δ-scores of the group are in the interval [0.933, 0.979], the difference needed to have the same average is $0.115\bar{\delta}$, which leaves our corrected $(\hat{\beta}_1^S + \hat{\beta}_2^S)(1 + \hat{\beta}_1^f)$ higher in the Zero-collusion group than it is in the Collusion group.

$$\begin{aligned} &2\beta_0^S + (\beta_1^S + \beta_2^S)\beta_0^f + \beta_2^f(\beta_1^S + \beta_2^S)\gamma \\ &= \begin{cases} 2 \cdot 0.425 + (1.398 - 1.139)0.271 = 0.920 & \text{Zero-collusion group} \\ 2 \cdot 0.478 + (1.275 - 1.107)0.444 + \underbrace{0.058(-1.107 + 1.275)}_{<.001}\gamma = 1.03 & \text{Collusion group} \end{cases} \end{aligned} \tag{4.2.10}$$

To sum up, we have shown that the introduction of non-myopic agents is not really jeopardizing the ability of our implementation of the DDPG algorithm to find an economically-valid and consistent equilibrium. If it can be slightly more disordered and no longer follows a normal law, these issues are in fact features as it shows the alternance between the regular equilibrium and a more collusive one. On the possible collusion, we find that γ has no direct impact on the collusive outcome, contrary to the optimal tuple formed by the duration of the simulation, the memory buffer length and the decay rate. We have shown that the fact that the tuple of these parameters is at the value $(m = 1000, T = 9 \times 10^4, \omega = .9999)$ impacts the collusiveness of the outcome by distorting the response to a higher relative valuation of the cooperative equilibrium, by reducing the weight of the betray temptation.

4.3. ***From duopolies to oligopolies in Cournot games: a journey among stable and chaotic equilibria***

In this section, our goal is to study the possible shifts in behaviors that can be observed when our markets are populated by more than two firms. The challenge is interesting as this configuration may be the most common in real life markets. The objective of this part is to study the consistency of obtained equilibria, the effects of non-myopic agents, and the behavior of the model with non-stable settings (which cause chaotic equilibria when studied analytically). Similarly to what have been done previously, we compute the tuple of parameters that will allow us to perform consistent simulations with our framework, i.e. simulations that will never require neither the critic nor the actor to respond with a higher value than one. We use $D = 5$, $\eta_\pi = .25$, $H_\pi = .15$, with $c = .6$ in "stable" simulations and $c = .5$ in unstable simulations. We compute the equilibrium tuple corresponding to these values: (0.962, 0.289, 0.806, 0.410, 0.543, 0.489) for the stable case, and (1, 0.275, 0.833, 0.410, 0.556, 0.497) for the unstable case[4]. All simulations are conducted with $T = 9 \times 10^4$ and with a linear exploration white noise decay process with maximum variance (ς) of .3.

Sample			Inter-dispersion			Distribution		Intra-dispersion			
			$Q^*_{\text{Co.}}$	$Q^*_{\text{Co.}}$	Shapiro						
c	γ	m	$\pm5\%$	$\pm10\%$	p-value	$\bar{Q}$	sd(Q)	mean($\breve{S}$)	sd($\breve{S}$)	$\bar{\Delta}$	$\#\mathcal{S}$
0.6	0	500	0.94	1	0.001	3.20	0.089	0.232	0.093	0.028	113
0.6	0	1000	0.90	1	0.001	3.15	0.079	0.193	0.086	0.120	97
0.6	0	2000	0.93	1	0.296	3.16	0.071	0.185	0.070	0.117	43
0.6	0.5	500	0.68	0.92	0.008	3.33	0.152	0.280	0.101	-0.258	84
0.6	0.5	1000	0.86	0.99	0.057	3.25	0.107	0.255	0.091	-0.081	83
0.5	0	500	0.94	1	0.122	3.34	0.096	0.240	0.083	-0.025	105
0.5	0	1000	1	1	0.230	3.31	0.074	0.203	0.070	0.034	30

TABLE 8. **Implementation performance of the DDPG Algorithm in the Cournot 4-oligopoly.**

We provide our usual indicators of what we have defined as intra and inter dispersion in Table 8. On the matter of unstable equilibria, our model seems not to be affected by the chaotic nature of the analytical equilibrium: in both settings, 94% of simulations have converged toward an equilibrium that is in the 5%-confidence interval. We find that our model has converged toward the same mean (after correcting for the effect of the change in c) with a p-value of .022, and, consequently, that the mean of our Δ-score are equal with a p-value of .049 (we have performed at each time a Welch t-test). These results are corroborated by the Kolmogorov-Smirnov test, that gives us a p-value of .8352, suggesting that the underlying distribution is the same for both samples. Nonetheless, if we cannot state that mean ($\breve{S}$) is significantly higher in the unstable group, we cannot state either that the two means are equal (p-value of .5).

We also perform the same simulations with non-myopic agents. If the simulations with non-myopic agents appear not consistent with $m = 500$ (the majority of convergence equilibria are above the Cournot analytical equilibrium, indicating that the agents have not found the optimal behavior), the group with $m = 1000$ exhibits solid convergence toward the Cournot equilibrium. We confirm a famous result in the literature about algorithmic collusion: the more firms are on the market, the less collusive behaviors are observed. We find by conducting a Welch t-test that we can safely reject the hypothesis that the means of the selected equilibrium ($\bar{Q}$) and the means of the Δ-scores are different in the sample ($\gamma = 0, m = 500$) and ($\gamma = .5, m = 1000$) (with p-values of .0005 and 0.0006). We also can state that the intra-dispersion is very close in both groups, with a Welch t-test p-value of .085. Overall, we can state that we find no evidence of an effect of the γ parameter on our simulations.

As we have previously noticed an effect of the memory length on the equilibrium, we perform the same simulations with different sizes for the memory buffer. We find that the behavior of the algorithm is not modified in the unstable case (all means are equal with p-value less than

[4]Obviously, this equilibrium is actually not a true equilibrium as we have shown in the Theoretical Framework that the model does not analytically converge toward a stable value. Here, we have computed the steady-state of the model, keeping in mind that the theory would impose that this value should never been durably observed as an equilibrium.

.08). Surprisingly, the stable case is more interesting: we find that the equilibrium reached decreased, indicating slight collusion (the Δ-score increases). Though the intra-dispersion appears constant (same mean with pvalue .002), the median dispersion is reduced with higher memory from .25 to .17, suggesting better convergence as the Nash-equilibrium should be symmetric. We find with very high statistical significance that the average Δ-score is higher in the high-memory group, and that the average quantity chosen is lower within the high-memory group (both p-values are equal to $4e^{-05}$). As previously noticed for the intra-dispersion, the inter-dispersion is also lower: we find with high significance (the p-value of the Kolmogorov-Smirmov test is .001) that both samples are not evenly distributed (after correcting for the difference in means), and that the high-memory group has a lower variance (with a Fisher test with p-value of .12, and a Levene test with p-value of .000). We do not notice any interest for increasing the memory length above $m = 1000$, as we do not observe any statistically significant differences with the group $m = 2000$ for instance (we could even affirm with more data that the underlying distribution is the same as the Kolmogorov-Smirmov gives a p-value of .867 indicating that the total quantity reached at the equilibrium is distributed following the same law between the two groups).

Despite noticing no major changes in the equilibrium reached between the stable setting and the unstable one, we point that both groups have converged toward an equilibrium that is highly asymmetrical, an effect that can be attributed directly to the shift from a duopoly to the 4-oligopoly. We find in each group an average intra-dispersion score of .2, where it was at .05 in the Cournot duopoly. Though the difference is important, we should keep in mind that our $\breve{S}$-score measures the maximum distance between agents and not the average, which in part justifies the dramatic increase of this indicator.

5. Discussion

5.1. ***Our results in the algorithmic-collusion field***

Overall, our results tend to support the thesis developed by Abada, Lambin, and Tchakarov 2022, especially in the Cournot 4-oligopolies. We can state that collusive outcomes have been very rare in our settings, and appear to be an exception caused by well-chosen parameters. When we increase learning time, and memory, we find, especially for the 4-oligopoly, that the algorithm converges toward the Nash equilibrium. Our only divergence with them (and with Graf et al. 2023) is on the memory parameter. We find that some combinations ($m = 1000$, $T = 90000$, $\omega = .9999$) allow the algorithm to find a collusive equilibrium that yields to agents a profit that is between 10 and 20 percentage points higher. This shows that a higher training does not necessarily leads to the Nash-equilibrium, and that the memory length does not only affect the time before convergence.

When we find collusion (in the case of the Cournot duopoly and in the Cournot 4-oligopoly), we do not spot any punishment behaviors as Calvano, Calzolari, and Denicolò 2019. Our results appear in perfect contradiction with the former approach (for instance for the influence of the forward looking γ parameter), suggesting that their results were implementation specific (Q-Learning, with their set of parameters and their ultra-short memory duration).

Overall, it seems that collusion in our settings has been observed when we reached ideal settings for the algorithms to collectively find better outcomes. The cases without any collusion are clearly more common than the ones where collusion can appear, and we highly doubt that the slight deviation from the equilibrium we observe can be replicated in real-life conditions, with non-identical algorithms, probably not optimally calibrated.

Works on algorithmic collusion often conclude on policy implications. Again, following the work of Abada, Lambin, and Tchakarov 2022, we do not find any reason to adjust anti-trust policies, as more sophisticated algorithms seem more likely to serve rational competition than collusion. However, despite our efforts to implement fully decentralized learning algorithms, we need to be aware that our models are not realistic as they assume fully identical algorithms, without pre-training and without temporal differences (all firms join the market at the same time). Such study could actualize overlooked previous works of the evolutionary game theory field which have studied the evolution of strategies in an heterogenous market (we pinpoint the work of Dixon, Wallis, and Moss 2002). Implementing heterogeneity, and question its implications on the obtained equilibrium, could make a very interesting work to allow these theoretical results to be more useful for policy-makers and corporate managers.

5.2. ***Implications on the validity of the dynamic Cournot model***

Our last section has allowed us to discover that our artificial market with self-learning agents are not significantly affected by market situations that are analytically unstable. We have found no evidence of a shift in behaviors. Worse, we find no element that could corroborate the best response adjustment process described by Cournot 1838 and studied analytically by Theocharis 1960 and his followers. We observe, as expected with a DDPG algorithm, a slow convergence toward a stable value, and not a converging (or diverging) oscillation around an equilibrium.

These results are confusing as they raise questions about the validity of the Cournot adjustment behavior. The lack of economical interpretation for this chaotic behavior, corroborated by our findings with artificial agents, should make us doubt of the validity of the model. What is surprising though, is the accuracy that characterize the converging behavior of our model toward the analytical steady state, even if this state should not be stable. This question requires now a theoretical investigation, to give new ground for this model that predicts well, but fail to explain why.

Conclusion

Our main contribution has been to introduce the first agent-based model of competition in quantities featuring a *Deep Deterministic Policy Gradient* (DDPG) algorithm. This algorithm has been selected as a replacement for the traditional Q-Learning algorithm that impose dramatic simplifications (discretization of the action set, ...), to examine two current unsolved questions in the economic literature: the tendency of algorithmic markets to converge toward a collusive equilibrium, and the chaotic behavior of the dynamic Cournot oligopoly.

To address these questions, we have shown that the DDPG algorithm, despite theoretical reserves, is a relevant tool to model multi-agent systems (MAS). We have exhibited that both in the Stackelberg and in the Cournot model, the DDPG algorithmic-agent was able to find the Nash equilibrium and, hence, to maximize its profit, without any knowledge on the other firms nor on the demand function. We have studied the effect of hyper-parameters on the behavior of the algorithm and find consistent results with the previous literature.

Then, we have shown that, under well-tuned parameters and circumstances, the DDPG algorithm could, as the Q-Learning does, lead to collusion in the Cournot oligopoly. We have found that this collusion was linked to a certain combination of the memory of the agent, the training duration and the learning rate decay factor, and not to its actualization rate (γ). We have developed an estimation strategy that allowed us to show that the former parameters were impacting the equilibrium by adjusting the balance between the temptation to betray and the interest to play cooperatively. Collusion has been shown to appear only in optimal settings, and not as a general phenomenon.

However, we can state that we have demonstrated, by studying the dynamical Cournot 4-oligopoly, that our DDPG model always converge toward the Nash equilibrium (or above if not sufficiently trained, or below in optimal settings with collusion), but never follows a chaotic path. This result proves that chaotic equilibria are not a feature of the model. This shadows the theoretical validity of the adjustment process of the dynamical Cournot model, suggesting that the chaotic behavior only exists on paper.

Code and data availability

Our implementation of the DDPG algorithm has been entirely made from scratch in C++. We provide a GitHub repository (https://github.com/Aldric-L/DDPG-Oligopolies-Simulator) where both the source code and the simulations can be found. The source code is licensed under Creative Commons Attribution 4.0 International.

References

[1] Ibrahim Abada, Xavier Lambin, and Nikolay Tchakarov. “Collusion by Mistake: Does Algorithmic Sophistication Drive Supra-Competitive Profits?” In: *Social Science Research Network* (2022). DOI: 10.2139/ssrn.4099361.

[2] Hamdy N. Agiza and Abdelalim A. Elsadany. “Nonlinear dynamics in the Cournot duopoly game with heterogeneous players”. In: *Physica A: Statistical Mechanics and its Applications* 320 (2003), pp. 512–524. ISSN: 0378-4371. DOI: 10.1016/S0378-4371(02)01648-5.

[3] Hamdy N. Agiza and Abdelalim A. Elsadany. “Chaotic dynamics in nonlinear duopoly game with heterogeneous players”. In: *Applied Mathematics and Computation* (2004). DOI: 10.1016/s0096-3003(03)00190-5.

[4] *Algorithms and Collusion: Competition Policy in the Digital Age*. Tech. rep. OECD, 2017.

[5] *Algorithms: How they can reduce competition and harm consumers*. Tech. rep. Competition and Markets Authority, 2021.

[6] John Asker, Chaim Fershtman, and Ariél Pakes. “Artificial Intelligence, Algorithm Design, and Pricing”. In: *AEA papers and proceedings* (2022). DOI: 10.1257/pandp.20221059.

[7] Stephanie Assad et al. “Algorithmic Pricing and Competition: Empirical Evidence from the German Retail Gasoline Market”. In: *Journal of Political Economy* 132.3 (2024), pp. 723–771. DOI: 10.1086/726906.

[8] Martino Banchio et al. *Adaptive Algorithms, Tacit Collusion, and Design for Competition.* 2022. DOI: 10.48550/arXiv.2202.05946.

[9] David P. Byrne and Nicolas de Roos. “Learning to Coordinate: A Study in Retail Gasoline”. In: *American Economic Review* 109.2 (Feb. 2019), pp. 591–619. DOI: 10.1257/aer.20170116.

[10] Emilio Calvano, Giacomo Calzolari, and Vincenzo Denicolò. “Artificial Intelligence, Algorithmic Pricing, and Collusion”. In: *American Economic Review* (2019). DOI: 10.2139/ssrn.3304991.

[11] Le Chen, Alan Mislove, and Christo Wilson. “An Empirical Analysis of Algorithmic Pricing on Amazon Marketplace”. In: *Proceedings of the 25th International Conference on World Wide Web*. WWW ’16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016, pp. 1339–1349. ISBN: 9781450341431. DOI: 10.1145/2872427.2883089. URL: https://doi.org/10.1145/2872427.2883089.

[12] Antoine Augustin Cournot. *Recherches sur les Principes Mathématiques de la Théorie des Richesses*. Économistes. Paris: Hachette, 1838.

[13] Huw D. Dixon, Steven Wallis, and Scott Moss. “Axelrod Meets Cournot: Oligopoly and the Evolutionary Metaphor”. In: *Computing in Economics and Finance* (2002). DOI: 10.1023/a:1020922214711.

[14] Christoph Graf et al. “Computational Performance of Deep Reinforcement Learning to Find Nash Equilibria”. In: *Computational Economics* (2023). DOI: 10.1007/s10614-022-10351-6.

[15] Cars Hommes, Marius I. Ochea, and Jan Tuinstra. *On the Stability of the Cournot Equilibrium: An Evolutionary Approach.* 2011. DOI: 11245/1.354907.

[16] Yann Kerzreho. “Spontaneous Collusion with Synchronous Learning and States”. MA thesis. Ecole Normale Supérieure Paris-Saclay, 2024.

[17] Vijay Konda and John Tsitsiklis. “Actor-Critic Algorithms”. In: *Proceedings of Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[18] Marek Lampart, Alžběta Lampartová, and Giuseppe Orlando. “On extensive dynamics of a Cournot heterogeneous model with optimal response”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 32.2 (Feb. 2022), p. 023124. ISSN: 1054-1500. DOI: 10.1063/5.0082439.

[19] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning.* 2015. DOI: 10.48550/arXiv.1509.02971.

[20] Michael L. Littman. “Markov games as a framework for multi-agent reinforcement learning”. In: *Proceedings of the International Conference on Machine Learning.* 1994. DOI: 10.1016/b978-1-55860-335-6.50027-1.

[21] Ryan Lowe et al. “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments”. In: *Proceedings of the Neural Information Processing Systems conference.* 2017. DOI: 10.48550/arXiv.1706.02275.

[22] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning.* 2013. DOI: 10.48550/arXiv.1312.5602.

[23] Tõnu Puu. “On the stability of Cournot equilibrium when the number of competitors increases”. In: *Journal of Economic Behavior and Organization* (2008). DOI: 10.1016/j.jebo.2006.06.010.

[24] Adrian Redder, Arunselvan Ramaswamy, and Holger Karl. *3DPG: Distributed Deep Deterministic Policy Gradient Algorithms for Networked Multi-Agent Systems.* 2022. DOI: 10.48550/arXiv.2201.00570.

[25] Samuel S. Shapiro and M. B. Wilk. “An Analysis of Variance Test for Normality (Complete Samples)”. In: *Biometrika* (1965). DOI: 10.1093/biomet/52.3-4.591.

[26] David Silver et al. “Deterministic Policy Gradient Algorithms”. In: *Proceedings of the International Conference on Machine Learning.* June 2014.

[27] Leigh Tesfatsion. “Modeling economic systems as locally-constructive sequential games”. In: *Journal of Economic Methodology* 24.4 (2017), pp. 384–409. DOI: 10.1080/1350178X.2017.1382068.

[28] Leigh Tesfatsion. “Agent-Based Computational Economics: Overview and Brief History (1st edition)”. In: *Artificial Intelligence, Learning, and Computation in Economics and Finance.* Ed. by Ragupathy Venkatachalam. Springer, 2023, pp. 41–58.

[29] Reghinos D. Theocharis. “On the Stability of the Cournot Solution on the Oligopoly Problem”. In: *The Review of Economic Studies* (1960). DOI: 10.2307/2296135.

[30] Ludo Waltman and Uzay Kaymak. “Q-learning agents in a Cournot oligopoly model”. In: *Journal of Economic Dynamics and Control* (2008). DOI: 10.1016/j.jedc.2008.01.003.

[31] Chris Watkins. “Learning from delayed rewards”. PhD thesis. 1989. URL: https://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

[32] Kaiqing Zhang et al. “Fully decentralized multi-agent reinforcement learning with networked agents”. In: *International Conference on Machine Learning* (2018). DOI: 10.48550/arXiv.1802.08757.

[33] Shiyang Zhou et al. “Independent Deep Deterministic Policy Gradient Reinforcement Learning in Cooperative Multiagent Pursuit Games”. In: *Proceedings of International Conference on Artificial Neural Networks.* 2021. DOI: 10.1007/978-3-030-86380-7_51.

Appendix A. The Cournot stability problem

In this appendix, we will prove that the Cournot static game is the limit of the converging case of the Cournot temporal model. We also determine a convergence criteria.

The produced quantities are determined by the following system of first-order difference equations:

$$\begin{cases} q_t^1 = \frac{D-\sum_{i\neq 1} q_{t-1}^i}{2(1+c)} \\ \vdots \\ q_t^N = \frac{D-\sum_{i\neq N} q_{t-1}^i}{2(1+c)} \end{cases} \Leftrightarrow \underbrace{\begin{bmatrix} q_t^1 \\ \vdots \\ q_t^N \end{bmatrix}}_{Q_t} = \frac{-1}{2(1+c)} \left(\underbrace{\begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \ddots & \vdots \\ 1 & 1 & \ddots & \ddots & 1 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} q_{t-1}^1 \\ \vdots \\ q_{t-1}^N \end{bmatrix}}_{Q_{t-1}} - \underbrace{\begin{bmatrix} D \\ \vdots \\ D \end{bmatrix}}_{B} \right) \tag{A.0.1}$$

This can be rewritten in the form of a sequence:

$$\begin{aligned} Q_t &= \frac{-1}{2(1+c)} \left(AQ_{t-1} + B\right) \\ &= \left(\frac{-1}{2(1+c)}\right)^2 A^2 Q_{t-2} + \left(\frac{-1}{2(1+c)}\right)^2 AB + \frac{-1}{2(1+c)} B \\ &\vdots \\ Q_t &= \left(\frac{-1}{2(1+c)}\right)^t A^t Q_0 + \sum_{0\leqslant k\leqslant n} \left(\frac{-1}{2(1+c)}\right)^{k+1} A^k B \end{aligned} \tag{A.0.2}$$

To compute A^t we perform spectral analysis on this $\mathcal{M}_N(\mathbb{R}), N > 2$:

$$\mathrm{Sp}(A) = \{-1, (N-1)\}$$

First, as $0 \notin \mathrm{Sp}(A)$ and as $1 \notin \mathrm{Sp}(A)$ we deduce that A and $A - I_n$ are non singular matrices. Hence, we can refine equation A.0.2:

$$\begin{aligned} Q_t &= \left(\frac{-1}{2(1+c)}\right)^t A^t Q_0 + \frac{-1}{2(1+c)} \left[\sum_{0\leqslant k\leqslant t} \left(\frac{-1}{2(1+c)}\right)^k A^k \right] B \\ &= \left(\frac{-1}{2(1+c)}A\right)^t Q_0 + \frac{-1}{2(1+c)} \left[\left(\left(\frac{-1}{2(1+c)}A\right)^{t+1} - I_N \right) \left(\frac{-1}{2(1+c)}A - I_N\right)^{-1} \right] B \\ &= \left(\frac{-1}{2(1+c)}A\right)^t \left[Q_0 + \left(\frac{-1}{2(1+c)}\right)^2 A \left(\frac{-1}{2(1+c)}A - I_N\right)^{-1} B \right] \\ &+ \frac{1}{2(1+c)} \left[\left(\frac{-1}{2(1+c)}A - I_N\right)^{-1} \right] B \end{aligned} \tag{A.0.3}$$

We simplify the latter expression (A.0.3) by denoting $k = \frac{-1}{2(1+c)}$:

$$Q_t = (kA)^t \left[Q_0 + k(kA)\left(kA - I_N\right)^{-1} B\right] - k\left(kA - I_N\right)^{-1} B \tag{A.0.4}$$

Proposition A.1. *for $N > 2$ we have $A = PDP^{-1}$ with*

$$D = \begin{bmatrix} N-1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \ddots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & -1 \end{bmatrix}, P = \begin{bmatrix} 1 & -1 & -1 & \cdots & -1 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & \ddots & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 1 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{A.0.5}$$

The last proposition allows us to rewrite A.0.4:

$$Q_t = P\,(kD)^t\,P^{-1}\left[Q_0 + k(kPDP^{-1})\left(kPDP^{-1} - I_N\right)^{-1}B\right] - k\left(kPDP^{-1} - I_N\right)^{-1}B \tag{A.0.6}$$

The latter equality provides a convergence criteria:

$$\lim_{t\to\infty}(kD)^t = 0_{\mathcal{M}_N(\mathbb{R})} \Leftrightarrow |k(N-1)| < 1$$
$$\Leftrightarrow \left|\frac{-1}{2(1+c)}\right| < \frac{1}{N-1} \Leftrightarrow \boxed{\frac{N-3}{2} < c} \text{ or } \frac{-(N+1)}{2} > c \tag{A.0.7}$$

Assume that this condition is met, we have:

$$\lim_{t\to\infty} Q_t = -k\,(kA - I_N)^{-1}\,B \tag{A.0.8}$$

We can now show that this limit is the standard Cournot model:

$$\lim_{t\to\infty} Q_t = -k\,(kA - I_N)^{-1}\,B = \left(\frac{1}{k}I_N - A\right)^{-1} B \tag{A.0.9}$$

Let us recall that the standard Cournot model is:

$$\begin{cases} q^1 = \frac{D-\sum_{i\neq 1} q^i}{2(1+c)} \\ \vdots \\ q^N = \frac{D-\sum_{i\neq N} q^i}{2(1+c)} \end{cases} \Leftrightarrow \underbrace{\begin{bmatrix} q^1 \\ \vdots \\ q^N \end{bmatrix}}_{Q^C} = \frac{-1}{2(1+c)}\left(\underbrace{\begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \ddots & \vdots \\ 1 & 1 & \ddots & \ddots & 1 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1 & 0 \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} q^1 \\ \vdots \\ q^N \end{bmatrix}}_{Q^C} - \underbrace{\begin{bmatrix} D \\ \vdots \\ D \end{bmatrix}}_{F}\right) \tag{A.0.10}$$

$$\Leftrightarrow Q^C = k(EQ^C + F) \Leftrightarrow Q^C = (k^{-1}I_N - E)^{-1}F \tag{A.0.11}$$

By noticing that $E = A$ and $F = B$, we have the fundamental result, that is, **if the convergence criterion A.0.7 is met, the two models are asymptotically equal**:

$$\boxed{\lim_{t\to\infty} Q_t = -k\,(kA - I_N)^{-1}\,B = Q^C} \tag{A.0.12}$$

Proof of equation A.0.5. We will first prove by induction that the characteristic polynomial of matrix A is $P_n^A(\lambda) = -(1+\lambda)P_{n-1}^A + (-(1+\lambda))^{n-1}$ for $n \geqslant 2$ with $P_1^A(\lambda) = -\lambda$:

Initialization: $P_2^A = \begin{vmatrix} -\lambda & 1 \\ 1 & -\lambda \end{vmatrix} = \lambda^2 - 1 = -(\lambda+1)(-\lambda) - (1+\lambda)^1$

Inheritance: Assume that $P_n^A(\lambda) = -(1+\lambda)P_{n-1}^A + (-(1+\lambda))^{n-1}$ for $n \geqslant 2$ with $P_1^A = -\lambda$ is true for a given $n \geqslant 2$.

$$P_{n+1}^A(\lambda) = \begin{vmatrix} -\lambda & 1 & 1 & \cdots & 1 \\ 1 & -\lambda & 1 & \cdots & 1 \\ 1 & 1 & \ddots & 1 & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 1 & 1 & \cdots & 1 & -\lambda \end{vmatrix} = \begin{vmatrix} -\lambda-1 & 0 & \cdots & 0 & 1+\lambda \\ 1 & -\lambda & 1 & \cdots & 1 \\ 1 & 1 & \ddots & 1 & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 1 & 1 & \cdots & 1 & -\lambda \end{vmatrix}_{L_1 \leftarrow L_1 - L_{n+1}}$$

$$= (-\lambda-1)\begin{vmatrix} -\lambda & 1 & \cdots & 1 \\ 1 & \ddots & 1 & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & -\lambda \end{vmatrix} + (-1)^n(1+\lambda)\begin{vmatrix} 1 & -\lambda & 1 & 1 \\ 1 & 1 & \ddots & 1 \\ \vdots & \vdots & \ddots & -\lambda \\ 1 & 1 & \cdots & 1 \end{vmatrix}$$

We can simplify the right hand side by an immediate recurrence with $L_1 \leftarrow L_1 - L_n$ such that:

$$\begin{vmatrix} 1 & -\lambda & 1 & 1 \\ 1 & 1 & \ddots & 1 \\ \vdots & \vdots & \ddots & -\lambda \\ 1 & 1 & \cdots & 1 \end{vmatrix} = \begin{vmatrix} 0 & -\lambda-1 & 0 & 0 \\ 1 & 1 & \ddots & -\lambda \\ \vdots & \vdots & \ddots & -\lambda \\ 1 & 1 & \cdots & 1 \end{vmatrix}_{L_1 \leftarrow L_1 - L_n} = (\lambda+1)\begin{vmatrix} 1 & -\lambda & 1 & 1 \\ 1 & 1 & \ddots & 1 \\ \vdots & \vdots & \ddots & -\lambda \\ 1 & 1 & \cdots & 1 \end{vmatrix} = \cdots = (1+\lambda)^n$$

Hence we can conclude and verify the inheritance:

$$P_{n+1}^A(\lambda) = -(1+\lambda)P_n^A + (-(1+\lambda))^n$$

Now that we have the characteristic polynomial of A, we can express it as a function of n and λ. We prove by induction that $\forall n \geqslant 1, P_n^A(\lambda) = (-1)^{n-2}(1+\lambda)^{n-1}(\lambda - n + 1)$.

Initialization: $P_1^A(\lambda) = -\lambda$.

Inheritance: Assume that $P_n^A(\lambda) = (-1)^{n-2}(1+\lambda)^{n-1}(\lambda - n + 1)$ for $n \geqslant 1$ with $P_1^A = -\lambda$ is true for a given $n \geqslant 1$. By the previous induction, we get:

$$\begin{aligned} P_{n+1}^A(\lambda) &= -(1+\lambda)(-1)^{n-2}(1+\lambda)^{n-1}(\lambda-n+1) + (-(1+\lambda))^n \\ &= (-1)^{n-1}(1+\lambda)^n(\lambda-n+1) + (-(1+\lambda))^n \\ &= (-1)^{n-1}(1+\lambda)^n(\lambda-n) \end{aligned}$$

Which completes the proof of the characteristic polynomial.

From the previous proof, we have that $\text{Sp}(A) = \left\{\lambda \in \mathbb{R}, P_n^A(\lambda) = 0\right\} = \{-1, n-1\}$. We can find the eigenvectors associated with these eigenvalues.

For -1: We first use the rank theorem $\text{rank}(A+I_n) + \dim(\ker(A+I_n)) = n \Rightarrow \dim(\ker(A+I_n)) = n-1$ as $(A+I_n)$ is the matrix whose every coefficient is one. Hence, we get that the dimension of the eigenspace associated with -1 is $n-1$. Let's determine a basis for this eigenspace. We observe that we are looking for every vector $u \in \mathbb{R}^n$ such that $(A + I_n)u = 0_{\mathbb{R}^n} \Rightarrow u \in \{u \in \mathbb{R}^n, \sum_{i\leqslant n} u_i = 0\}$. With e_i the canonical vector of $\mathbb{R}^n$, we propose the family $(e_1-e_2, e_2-e_3, ..., e_{n-1}-e_n)$ which is free by construction and hence a basis of $\{u \in \mathbb{R}^n, \sum_{i\leqslant n} u_i = 0\}$. We conclude that $E_{-1} = \text{vect}(e_1 - e_2, e_2 - e_3, ..., e_{n-1} - e_n)$.

For $n-1$: By a similar argument, we are looking for the dimension of $\ker(A + (1-n)I_n)$.

$$\begin{aligned}
\text{rk}(A+(1-n)I_n) &= \text{rk}\begin{bmatrix} 1-n & 1 & 1 & \cdots & 1 \\ 1 & 1-n & 1 & \ddots & \vdots \\ 1 & 1 & \ddots & \ddots & 1 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1 & 1-n \end{bmatrix} \\
&= \text{rk}\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 1-n & 1 & \cdots & 1 \\ 1 & 1 & 1-n & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1 & 1-n \end{bmatrix}_{L_1 \leftarrow \sum_{0<i\leqslant n} L_i} \\
&= \text{rk}\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ n-1 & -1 & -1 & \cdots & -1 \\ 1 & 1 & 1-n & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1 & 1-n \end{bmatrix}_{L2 \leftarrow \sum_{1<i\leqslant n} L_i} \\
&= \text{rk}\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ n-1 & -1 & -1 & \cdots & -1 \\ n & 0 & -n & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ n & 0 & \cdots & 0 & -n \end{bmatrix}_{\forall i\in[3,n], L_i \leftarrow L_i + L_2} \\
&= \text{rk}\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & -1 & 0 & \cdots & 0 \\ n & 0 & -n & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ n & 0 & \cdots & 0 & -n \end{bmatrix}_{L_2 \leftarrow L_2 - \frac{1}{n}\sum_{2<i\leqslant n} L_i} \\
&= \text{rk}\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -1 & 0 & \cdots & 0 & 1 \end{bmatrix}_{\forall i\in[3,n], L_i \leftarrow \frac{-1}{n} L_i}
\end{aligned}$$

Hence we have that $\text{rk}(A + (1-n)I_n) = n - 1$, and by the rank theorem, we have $\dim(\ker(A + (1-n)I_n)) = 1$ We propose the family formed by the vector $(n-1, 0, ..., 0)$ as a basis for $\{u \in \mathbb{R}^n, (A + (1+n)I_n)u = 0_{\mathcal{M}_n}\}$. As the dimension of this family is one, and the dimension of the eigenspace is one, as the vector belongs to the eigenspace and as the family is trivially free, we have that: $E_{n-1} = \text{vect}((n-1, 0, ..., 0))$. □