DICE

DISCUSSION PAPER

NO 370

Collusion among Autonomous Pricing Algorithms Utilizing Function Approximation Methods

Malte Jeschonneck

August 2021

hhu
Heinrich Heine
Universität Düsseldorf

IMPRINT

DICE DISCUSSION PAPER

Published by:
Heinrich-Heine-University Düsseldorf,
Düsseldorf Institute for Competition Economics (DICE),
Universitätsstraße 1, 40225 Düsseldorf, Germany
www.dice.hhu.de

Editor:
Prof. Dr. Hans-Theo Normann
Düsseldorf Institute for Competition Economics (DICE)
Tel +49 (0) 211-81-15125, E-Mail normann@dice.hhu.de

ISSN 2190-9938 (online) / ISBN 978-3-86304-369-8

Collusion among Autonomous Pricing Algorithms Utilizing Function Approximation Methods

Jeschonneck, Malte *

August 2021

Abstract

The increased prevalence of pricing algorithms incited an ongoing debate about new forms of collusion. The concern is that intelligent algorithms may be able to forge collusive schemes without being explicitly instructed to do so. I attempt to examine the ability of *reinforcement learning* algorithms to maintain collusive prices in a simulated oligopoly of price competition. To my knowledge, this study is the first to use a reinforcement learning system with linear function approximation and eligibility traces in an economic environment. I show that the deployed agents sustain supra-competitive prices, but tend to be exploitable by deviating agents in the short-term. The price level upon convergence crucially hinges on the utilized method to estimate the qualities of actions. These findings are robust to variations of parameters that control the learning process and the environment.

I thank Hans-Theo Normann for providing many useful comments and encouraging me to publish this study. I was granted access to the High-Performance-Computing cluster *HILBERT* for which I am grateful.
Contact Information: E-mail: Malte.Jeschonneck@hhu.de

1. Introduction

There is little doubt that algorithms will play an increasingly important role in economic life. Dynamic pricing software is just one of several applications that already impacts market activities. Its frequent usage is reported in online retail markets (Chen, Mislove, and Wilson 2016), the tourist industry (Boer 2015) and petrol stations (Assad et al. 2020). As with many other technological advances, the economic advantages are conspicuous. Not only does automating pricing decisions cut costs and free up resources, algorithms may also be better at predicting demand and react faster to changing market conditions (OECD 2017). Overall, it seems likely that pricing algorithms may be used as a tool by companies to gain competitive advantages. It is worth pointing out that thereby intensified competition also benefits consumers.[1]

Nevertheless, concerns have been raised that ceding pricing authority to algorithms has the potential to create new forms of collusion that contemporaneous competition policy is not well equipped to deal with.[2] As Harrington (2018) notes, collusion itself is not considered illicit behavior. Rather, it is the process, by which it is achieved, that determines legality. Explicit communication among competitors who consciously agree on price levels is illegal. Smart adaption to market conditions by individual agents is typically tolerated despite the economic effect on consumers being equally detrimental (Motta 2004). In practice, the distinction is sometimes vague and the advent of pricing algorithms is believed to blur the line. If and when cooperating algorithms could elude competition enforcement is not immediately clear and subject to ongoing debate.[3]

It is yet to be seen how likely the scenario of algorithms achieving collusion in real markets is. Indeed, if it turns out to be improbable, any liability considerations are rendered superfluous. Unfortunately, the empirical evidence is scarce. An exception is a notable study of the German retail gasoline market by Assad et al. (2020) who document

[1] Of course other types of algorithms that benefit consumers exist. Price comparison tools have been around for a while but applications extend beyond a mere reduction of search costs. Gal and Elkin-Koren (2017) champion *algorithmic consumers*, electronic assistants that completely take over purchase decisions and may challenge market power of suppliers by bundling consumer interests.

[2] A different issue is that pricing algorithms with information on consumer characteristics may be able to augment the scope of *price discrimination*, i.e. companies extracting rent by charging to every consumer the highest price he or she is willing to pay. Under which circumstances competition authorities should be concerned with this possibility is outlined in OECD (2016). Ezrachi and Stucke (2017) develop a scenario where discriminatory pricing and tacit collusion occur simultaneously. Both issues remain outside the scope of this study.

[3] Academics seem to consent that algorithms could be utilized to facilitate *existing* collusive agreements (OECD 2016, Ezrachi and Stucke 2018). While these scenarios may alter the operational scope of market investigations, they are well covered by contemporary competition practices (Bundeskartellamt 2021, refer to CMA 2016 and oefgen 2019 for two exemplary cases). The more controversial scenario concerns independently developed or acquired algorithms that align pricing behavior (Mehra 2015, Ezrachi and Stucke 2017, Ittoo and Petit 2017, Harrington 2018, Schwalbe 2018, Gal 2019). See DoJ (2017) and EU (2017) for institutional statements.

that margins in duopoly markets increased substantially after both duopolists switched from manual pricing to algorithmic-pricing software. Further field studies could prove instrumental to confirm and refine these findings.

As a complement to empirical evidence, there is a growing number of simulation studies that show the capability of *reinforcement learning* algorithms to create and sustain collusive equilibria in repeated games of competition.[4] A seminal study by Waltman and Kaymak (2008) examines two *Q-Learning* agents in a *Cournot* environment. Their simulations result in supra-competitive outcomes. However, even memoryless agents without knowledge of past outcomes manage to attain quantities below the one-shot Nash equilibrium. This casts doubt on the viability of the learned strategies vis-à-vis rational agents. Truly memoryless agents can not pursue punishment strategies because they are unable to even detect them. Thus, constantly playing the one-shot solution *should* be the only rational strategy. It appears, the agents do not learn how to collude, but rather fail to learn how to compete. Kimbrough and Murphy (2009) trial a *probe and adjust* algorithm inspired by the management literature. They find that agents end up playing one-shot Nash prices unless industry profits enter the reward function in some way.[5]

Recent studies have focused on price instead of quantity competition. In a groundbreaking effort, Calvano et al. (2020) show that Q-Learning agents learn to sustain collusion through a *reward-punishment* scheme in a simultaneous pricing environment. These findings are remarkably robust to variations and extensions. Furthermore, they find that agents learn to price competitively if they are memoryless (i.e. can not remember past prices) or short-sighted (i.e. do not value future profits). This coincides with predictions from economic theory. An important extension comes from Hettich (2021). Instead of a Q-Learning algorithm, he utilizes function approximation, specifically a *deep Q-Network algorithm* originally due to Mnih et al. (2015). He shows that the method converges much faster than *Q-Learning*. The importance of that finding is augmented by the fact that the algorithm is much easier to scale to real applications.[6]

Lastly, Klein (2021) shows that *Q-Learning* algorithms in a *sequential* price setting en-

[4]The literature on the general behavior of learning algorithms in cooperative and competitive multi-agent games concerns a variety of applications and is impressively sophisticated (see e.g. Leibo et al. 2017 and Crandall et al. 2018 for recent large-scale experimental studies.) In comparison, their application in oligopolistic environments has been rare and the trialed algorithms have been relatively simple.

[5]To my knowledge, *probe and adjust* is the only algorithm that explored continuous price setting in repeated games of competition to date. The agents repeatedly draw from a confined, but continuous price range. After some interval, they assess whether low or high prices yielded better rewards and adjust the range of prices accordingly.

[6]Johnson, Rhodes, and Wildenbeest (2020) provide another supplement. Introducing a *multi-agent reinforcement learning* approach, they show that collusion arises even when the number of agents, often regarded a main inhibitor of cooperative behavior, is significantly increased. Moreover, they show that market design can disturb collusion.

vironment maintain a supra-competitive price level. He reports two types of equilibria: constant market prices and *Edgeworth price cycles* where competitors sequentially undercut each other until profits become low and one firm resets the cycle by increasing its price significantly.[7] Importantly, the high price levels are underpinned by a *reward-punishment scheme*, i.e. a price cut of one agent evokes punishment prices by the opponent. Interestingly, the agents return to pre-deviaton levels within a couple of periods.

In summary, recent simulation studies show that reinforcement learning algorithms are capable of colluding in prefabricated environments. However, most of them use a simple tabular learning method, called *Q-Learning*, that requires discretizing prices and does not scale well if the complexity of the environment increases (Ittoo and Petit 2017). Therefore, the direct transferability of these findings to real markets is questionable. Similar to Hettich (2021), this study attempts to mitigate these problems by employing *linear function approximation* to estimate the value of actions. More specifically, I develop three methods of function approximation and run a series of experiments to assess how they compare to tabular learning. Moreover, I utilize *eligibility traces* as an efficient way to increase the memory of agents interacting in the environment.[8]

To preview the results, the simulations show that the developed function approximation methods, like tabular learning, result in supra-competitive prices upon convergence. However, *unlike* tabular learning, the learned strategies are easy to exploit. By forcing one of the agents to diverge from the convergence equilibrium, I show that the cheated agent fails to punish that deviation. This indicates that the learned equilibrium strategies are unstable vis-à-vis rational agents with full information. This observation is robust to a number of variations and extensions. Also, with respect to eligibility traces, excessively increasing memory tends to destabilize the learning process, but the overall impact for reasonable parameter values appears small.

The remainder of this paper is organized as follows. The next section introduces the repeated pricing environment in which the artificial competitors interact and presents in detail the deployed learning algorithm with its parametrization. I present the results in section 3 and consider variations and extensions in section 4. Section 5 concludes.

[7] Noel (2008) considers a similar environment. However, he uses *dynamic programming* for learning. His deployed agents *know* their environment in detail, an assumption unlikely to hold in real markets. With *Q-Learning*, agents estimate the action values based on past experiences.

[8] Neither linear function approximation nor eligibility traces are new concepts in reinforcement learning. However, to my knowledge, this is the first study to apply them to a repeated pricing game.

2. Environment and learning algorithm

This section begins with brief a presentation of the simulated economic environment that the autonomous pricing agents interact with. Then I describe in detail the learning algorithm utilized by the agents and the methods to approximate the value of state-action combinations.

2.1. Economic environment

I consider an infinitely repeated pricing game with a multinominal logit demand as in Calvano et al. (2020). Restricting the analysis to a symmetric oligopoly case with $n = 2$ agents (where $i = 1, 2$), the market comprises two differentiated products and an outside option. In every period t, both agents simultaneously pick a price p_i. Demand for agent i is then determined:[9]

$$q_{i,t} = \frac{e^{\frac{a-p_{i,t}}{\mu}}}{\sum_{j=1}^{n} e^{\frac{a-p_{j,t}}{\mu}} + e^{\frac{a_0}{\mu}}} \tag{1}$$

The parameter μ controls the degree of horizontal differentiation, where $\mu \to 0$ approximates perfect substitutability. In this study I forego to incorporate vertical differentiation. But it is easily incorporated by choosing firm-specific quality parameters a. a_0 reflects the appeal of the outside good. It diminishes as $a_0 \to -\infty$. Profits of both agents π_i are simply calculated as

$$\pi_{i,t} = (p_{i,t} - c)q_{i,t} \quad , \tag{2}$$

where c is the marginal cost. Anderson and Palma (1992) show that the multinominal logit demand model with symmetric firms entails a unqique one-shot equilibrium with best responses that solve:

$$p_n = p^* = c + \frac{\mu}{1 - (n + e^{\frac{a_0 - a + p^*}{\mu}})^{-1}} \tag{3}$$

Naturally, the other extreme, a collusive (or monopoly) solution, is obtained by maximizing joint profits. Both, the Nash outcomes characterized by p_n and π_n *and* the fully collusive solution (p_m and π_m) shall serve as benchmarks for the simulations.

[9] The model was pioneered by Anderson and Palma (1992). Generalization to a model with n agents is straightforward. In fact, the demand formula remains the same. The limitation to 2 agents is merely chosen for computational efficiency and the (intuitive) conjecture that the simulation results generalize to more players.

Market entry and exit are not considered. The parametrization is identical to the baseline in Calvano et al. (2020): $c = 1$, $a = 2$, $a_0 = 0$ and $\mu = \frac{1}{4}$. These parameters give rise to a static Nash equilibrium with $p_n \approx 1.47$ and $\pi_n \approx 0.23$ per agent. The fully collusive solution engenders $p_m \approx 1.92$ with $\pi_m \approx 0.34$.

2.2. Learning algorithm

This section describes the way both players approach, interact with and learn from the environment presented in section 2.1. For the sake of simplicity, I will present that process from the vantage point of a single player. Accordingly, the subscript i is dropped when appropriate. The player's objective is to maximize its net present value of discounted future profits:

$$max \sum_{t=0}^{\infty} \gamma^t \pi_t, \tag{4}$$

where $\gamma \in [0, 1]$ is the discount factor.[10] Importantly, the decision maker does not know about the demand function, its opponent's pricing strategy or the benchmark values p_m and p_n. Rather, the player can only set a price p and *retrospectively* observe the opponent's action and own profits. This paradigm can be formalized as a *Markov Decision Process* (MDP). In a MDP, at any point in time t, the player, or *agent*, observes the current state of the environment $S_t \in \mathcal{S}$ and proceeds to select an action $A_t \in \mathcal{A}$.[11] $\mathcal{S}$ and $\mathcal{A}$, respectively, represent the entirety of possible states and actions. The environment returns a reward R_t to the agent and moves to the next stage S_{t+1}.[12] In this study, the action set comprises a finite number of prices that are inputted into (1). The state set simply represents the actions (i.e. prices) of the previous period. The next section details the grid of available prices.

2.2.1. Price grid

For the trialed algorithms, it is necessary to discretize the action space. Compared to the baseline specification in Calvano et al. (2020), I consider a wider price range confined by

[10] There is a small notational predicament. In economic papers, δ usually represents the discount factor. However, reinforcement learning texts reserve δ for the *temporal difference error* (see section 2.2.4). Here, I will follow the latter convention and let γ denote the discount factor.

[11] From the agent's point of view, the *environment* consists of everything that is outside of its control, mainly the demand function *and* the behavior of the other agent.

[12] Sometimes, the reward due to A_t and S_t is modeled as R_{t+1} instead. The provided description of MDPs is very brief and specific to this study. For a more general treatment of MDPs, consult Sutton and Barto (2018). Calvano et al. (2020) and Hettich (2021) apply the MDP framework to the Bertrand environment in a more rigorous fashion.

a lower bound A^L and an upper bound A^U:

$$A^L = c \tag{5}$$

$$A^U = p_m + \zeta(p_n - c) \tag{6}$$

The lower bound ensures positive margins. It is conceivable that a human manager could implement a sanity restriction like that before ceding pricing authority to an algorithm.[13] The parameter ζ controls the extent to which the upper bound A^U exceeds the monopoly price. With $\zeta = 1$, the difference between A^L and p_n is equal to the difference between A^U and p_m. The available set of prices $\mathcal{A}$ is then evenly spaced out in the interval $[A^L, A^U]$:

$$\mathcal{A} = \{A^L, A^L + \frac{1(A^U - A^L)}{m-1}, A^L + \frac{2(A^U - A^L)}{m-1}, ..., A^L + \frac{(m-2)(A^U - A^L)}{m-1}, A^U\}, \tag{7}$$

where m determines the number of feasible prices. Note that the discretization implies that agents will not be able charge exactly p_n or p_m. However, by choosing m appropriately, one can get close (see section 2.2.5). Following Sutton and Barto (2018), I denote any possible action as a and the actual realization at time t as A_t.

In this simulation, the state set merely comprises two variables, namely the actions taken at $t-1$:

$$S_t = \{p_{i,t-1}, p_{j,t-1}\} \tag{8}$$

Accordingly, for both state variables, the set of possible states is identical to the feasible actions $\mathcal{A}$. However, this is not required with function approximation methods. Theoretically, any state variable could be continuous and unbounded. Similarly to actions, s denotes any possible state set and S_t refers to the actual states at t.

2.2.2. Value approximation

As established, the agent chooses an action based on the current state without knowing about the demand function. With the profit maximization objective from (4) in mind, how does the agent decide on which action to play? It considers all the information that is available, i.e. s, and estimates the value of playing a. With function approximation, a set of parameters $\boldsymbol{w}_t = \{w_{t,1}, w_{t,2}, ..., w_{t,D}\}$ maps any combination of S_t and A_t to a value

[13] Johnson, Rhodes, and Wildenbeest (2020) do the same.

estimate $\hat{q}_t$.[14] Formally:

$$\hat{q}_t = \hat{q}(S_t, A_t, \boldsymbol{w}_t) = \hat{q}(p_{i,t-1}, p_{j,t-1}, p_{i,t}, \boldsymbol{w}_t) \tag{9}$$

More specifically, each parameter w_d is associated with a counterpart x_d, called a *feature*. The value of every feature is derived from a state variable, the considered action or a combination thereof: $x_d = x_d(S_t, A_t)$. Taken together, the features form a vector $\boldsymbol{x}_t = \boldsymbol{x}(S_t, A_t) = \{x_1(S_t, A_t), x_2(S_t, A_t), ..., x_D(S_t, A_t)\}$. Any state-action combination can be represented by such a *feature vector*. It is important to realize that $\boldsymbol{x}_t$ is determined solely by the combination of s and a. The mechanism, by which the state-action combinations are mapped to numerical values remains constant over time. Consequently, there is no need to subscript x_d with respect to t. Contrary $\boldsymbol{w}_t$ constitutes the agent's valuation strategy at t which is continuously refined over time.

Note that I will only consider linear functions of $\hat{q}$. In this specific case, (9) can be written as the inner product of the feature vector and the set of parameters:

$$\hat{q}_t = \boldsymbol{x}_t \top \boldsymbol{w}_t = \sum_{d=1}^{D} x_d(S_t, A_t) w_d = \sum_{d=1}^{D} x_d(p_{i,t-1}, p_{j,t-1}, p_{i,t}) w_d \tag{10}$$

An intuitive policy to achieve profit maximization would be to always play the action a that maximizes the estimated value $\hat{q}$ given s. Indeed, this strategy, called a *greedy* policy, would be optimal if the values of state-action combinations were estimated perfectly at all times. However, initially, the agent has never interacted with the environment and its valuation of state-action combinations is necessarily naive. To improve the value estimation, it is necessary to *explore* the merit of various actions. The next section describes how the agent mixes between *exploration and exploitation*. Subsequently, section 2.2.4 describes how the agent continuously learns to improve the set of parameters $\boldsymbol{w}$ from interacting with the environment and, ultimately, refines its value estimation.

2.2.3. Exploration and exploitation

In every period, the agent chooses either to *exploit* its current knowledge and pick the supposedly optimal action or to *explore* in order to test the merit of alternative choices that are perceived sub-optimal but may turn out to be superior. As is common, I use a

[14] In the computer science literature, $\boldsymbol{w}$ is typically referred to as *weights*. I will stick to the economic vocabulary and declare $\boldsymbol{w}$ parameters. Henceforth, I will use $d \in \{1, 2, ..., D\}$ to iterate over elements in $\boldsymbol{w}$. Moreover, from now one the time subscript is implied for individual components of $\boldsymbol{w}_t$, i.e. $w_{t,d} = w_d$.

simple ϵ-greedy policy to steer this tradeoff:

$$A_t = \begin{cases} arg\ \underset{a}{max}\ \hat{q}(S_t, a, \boldsymbol{w}_t) & \text{with probability } 1 - \epsilon_t \\ \text{randomize over } \mathcal{A} & \text{with probability } \epsilon_t \end{cases} \tag{11}$$

In words, the agent chooses to play the action that is regarded optimal with probability $1 - \epsilon_t$ and randomizes over all prices with probability ϵ_t. The subscript suggests that exploration varies over time. The explicit definition is equivalent to Calvano et al. (2020):

$$\epsilon_t = e^{-\beta t}\ , \tag{12}$$

where β is a parameter controlling the speed of decay in exploration. This *time-declining* exploration rate ensures that the agent randomizes actions frequently at the beginning of the simulation and stabilizes its behavior over time.

After both agents select an action, the quantities and profits are realized in accordance with (1) and (2). The agents' actions in period t become the state set in $t + 1$ and new actions are chosen again as dictated by (9) and (11).

Whether the agent decided to explore or to exploit, it proceeds to leverage the observed outcomes to refine $\boldsymbol{w}$.

2.2.4. Parameter update

After observing the opponent's price and own profits, the agent exploits this new information to improve its estimation technique. This study's full system to update the parameter vector $\boldsymbol{w}$ at t comprises the calculation of a *temporal-difference error* (TD error, denoted δ_t), a vector of eligibility traces $\boldsymbol{z}_t$ tracking the importance of individual coefficients in $\boldsymbol{w}_t$ and the final update rule to refine $\boldsymbol{w}_t$. The formulas are:

$$\delta_t = R_t + \gamma \bar{V}_t(S_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t)\ \ , \tag{13}$$

$$\boldsymbol{z}_t = \gamma\lambda\rho_t \boldsymbol{z}_{t-1} + \frac{\Delta \hat{q}_t}{\Delta \boldsymbol{w}_t}\ \ , \tag{14}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha\delta_t \boldsymbol{z}_t \tag{15}$$

I will explain all three of the system's components. Starting with (13), δ_t represents the so called *temporal-difference* or *TD error*, i.e. the error associated with the estimation of $\hat{q}_t$.[15] It measures the difference between the *ex ante* ascribed value to the selected state-action

[15]The utilized update system is referred to as *Expected SARSA*, where *SARSA* abbreviates a state-action-reward-state-action sequence.

combination in t on the right-hand side and the *ex post* actual reward in conjunction with the estimated value of the newly arising state-action combination in $t+1$ on the left-hand side. To elaborate on the latter, the reward $R_t = \pi_t - p_n$ reflects the profits relative to the Nash solution.[16] γ is the discount factor and is applied to the expected value of the upcoming state $\bar{V}_t(S_t)$, i.e. the average of all state-action estimates weighted by their probability of being played. Formally:

$$\bar{V}_t(S_{t+1}, \boldsymbol{w}_t) = \sum_a Pr(a|S_{t+1}, \boldsymbol{w}_t)\ \hat{q}(S_{t+1}, a, \boldsymbol{w}_t) \quad , \tag{16}$$

where $Pr(a|S_{t+1})$ represents the probability of selecting action a conditional on S_{t+1} in accordance with the ϵ-greedy policy from (11). A positive δ_t indicates that the value of playing A_t turned out to exceed the original expectation. Likewise, a negative δ_t suggests that the realization failed short of the estimated value. In both instances, $\boldsymbol{w}$ will be adjusted accordingly, such that the state-action combination is valued respectively higher or lower next time.

The second component $\gamma\lambda\rho_t \boldsymbol{z}_{t-1} + \frac{\Delta \hat{q}_t}{\Delta \boldsymbol{w}_t}$, labeled the eligibility trace $\boldsymbol{z}_t$, keeps track of the importance of individual coefficients in $\boldsymbol{w}$ when estimating $\hat{q}_t$ and, ultimately, selecting an action. The idea is that the eligibility trace controls the magnitude by which individual parameters are updated, prioritizing those that contributed to producing an estimate of $\hat{q}_t$. Though there exist various forms, (14) employs the most popular variation, called an *accumulating eligibility trace* (Sutton 1988). Specifically, the update comprises two components: a decay term $\gamma\lambda\rho_t \boldsymbol{z}_{t-1}$ and the gradient of $\hat{q}_t$. With respect to the former, note that $\boldsymbol{z}_{t-1}$ is multiplied with the known discount factor γ, another parameter $\lambda \in [0, 1]$ (more on this in section 2.2.5) and the *importance sampling ratio* ρ_t. ρ_t indicates whether the played action coincides with the strategy currently considered *optimal*. More specifically:

$$\rho_t = \frac{\kappa(A_t|S_t)}{Pr(A_t|S_t)} \quad , \tag{17}$$

where $Pr(A_t|S_t)$ is the probability of having selected A_t given S_t under the utilized ϵ-greedy policy and $\kappa(A_t|S_t)$ is the probability of selecting A_t given S_t under a hypothetical *target policy* without exploration ($\epsilon = 0$). Accordingly, ρ_t is zero if the agent chooses to explore a non-optimal action because, under the target policy, the probability of choosing

[16] Please note that I explicitly distinguish profits and rewards. Profits, π, represent the monetary remuneration from operating in the environment and can be interpreted economically. However, profits do not enter the learning algorithm directly. Instead, they serve as a precursor of rewards, R_t. Rewards constitute the signal that agents interpret as direct feedback to refine their algorithms.

a non-greedy action is null. On the other hand, ρ_t exceeds 1 if a greedy action is selected, because under the target policy the selected action is always greedy. In summary, the left term resets $\boldsymbol{z}$ to $\mathbf{0}$ once a non-greedy action is selected. Otherwise the specific combination of γ, λ and β (which affects $Pr(A_t|S_t)$) determines whether the components of $\boldsymbol{z}$ decay towards zero or accumulate.

The second term is the gradient of $\hat{q}$ with respect to $\boldsymbol{w}_t$:

$$\frac{\Delta \hat{q}_t}{\Delta \boldsymbol{w}_t} = \{\frac{\Delta \hat{q}_t}{\Delta w_1}, \frac{\Delta \hat{q}_t}{\Delta w_2}, ..., \frac{\Delta \hat{q}_t}{\Delta w_d}\} = \frac{\Delta(\boldsymbol{x}_t \top \boldsymbol{w}_t)}{\Delta \boldsymbol{w}_t} = \boldsymbol{x}_t \tag{18}$$

The final simplification is possible because I only consider approximations that are linear in parameters. (18) also elucidates the purpose of the eligibility trace. It puts higher weights on coefficients that have been used to decide on A_t and ensures that they can be altered faster and beyond the horizon of a single time period. To illustrate the benefits of eligibility traces consider the following example. An agent selects an action that yields a good immediate reward, but unexpectedly disappointing outcomes in the subsequent period. Without eligibility traces, the action yields a positive δ_t and the responsible coefficients in $\boldsymbol{w}$ are updated such that the action will be estimated to be more valuable next time. However, the method can not assign 'negative credit' to elements in $\boldsymbol{w}$ for the underwhelming outcomes in more distant future periods. In the long run, the agent might still end up refraining from playing the myopic action by taking into account the low estimated value of the subsequent state, but this might take several revisits to the state-action combination.

Eligibility traces can accelerate the learning process. While a similar immediate effect on updating $\boldsymbol{w}$ takes place, the eligibility trace 'remembers' which coefficients were adjusted in the past and enables a retrospective readjustment prompted by underwhelming rewards in future periods. Section 2.2.5 discusses that λ controls the degree of hindsight.

Armed with the TD error δ_t and the eligibility trace $\boldsymbol{z}_t$, the final piece of the system, (15), is an update rule to improve the parameter vector: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha\delta_t\boldsymbol{z}_t$. It comprises $\boldsymbol{w}_t$ (the current knowledge of the agent we hope to improve) and three components, α, δ_t and $\boldsymbol{z}_t$ that dictate the specific shape of the update. I will briefly discuss each parameter's role. The TD error δ_t specifies the direction and magnitude of the update. Recall that it estimates the error associated with the value estimation through $\hat{q}_t$. For instance, a δ_t close to 0 suggests the value of the state-action combination was estimated accurately and the update to $\boldsymbol{w}$ will be small. Contrary, a high (or strongly negative) δ unveils a large divergence between the agent's value estimation and the realized reward R_t and

warrants a more significant update to the parameters. The eligibility trace $\boldsymbol{z}_t$ specifies which elements in $\boldsymbol{w}$ are adjusted giving priority to *evocative* parameters. As discussed, $\boldsymbol{z}$ memorizes which parameters were responsible for the selected actions in the past and ensures a retrospective adjustment beyond a one-period horizon. Lastly, α steers the speed of learning. In this study, it is constant over time. I will briefly discuss selecting an appropriate value in section 2.3.2. The outlined reinforcement learning algorithm in this study, adapted from Sutton and Barto (2018) is summarized as Algorithm 1. Note that the execution stops once *convergence* is achieved. I describe the employed convergence rules in section 3.1.

Algorithm 1 Expected SARSA with eligibility traces.

input feasible prices via $m \in \mathbb{N}$ and $\zeta \geq 0$
configure static algorithm parameters $\alpha > 0$, $\beta > 0$, and $\lambda \in [0, 1]$
initialize parameter vector and eligibility trace $\boldsymbol{w}_0 = \boldsymbol{z}_0 = \mathbf{0}$
declare convergence rule (see section 3.1)
randomly initialize state S
start tracking time: $t = 1$
while convergence is not achieved, **do**
  select action A according to (11)
  observe profit π, adjust to reward R
  observe next state: $S_{t+1} \leftarrow A$
  calculate TD-error (13): $\delta \leftarrow R + \gamma \bar{V}(S_{t+1}) - \hat{q}(S_t, A)$
  update eligibility trace (14): $\boldsymbol{z} \leftarrow \gamma\lambda\rho\boldsymbol{z} + \boldsymbol{x}$
  update parameter vector (14): $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\delta\boldsymbol{z}$
  move to next stage: $S_t \leftarrow S_{t+1}$ and $t \leftarrow t + 1$
end while

2.2.5. Baseline parametrization

The presented learning paradigm requires specifying a number of parameters. Optimizing the setting is not a primary concern of this study. Nevertheless, I attempt to choose reasonable values for all parameters. To some degree, this task can be guided by theoretical considerations and previous simulation studies. Still, there remains a high degree of arbitrariness. To make the impact of those choices more transparent, section 4 will present results for other specifications. Table 1 presents the baseline parametrization of the learning algorithm as well as the considered variations. The discussion on appropriate values of the learning rate parameter α is delayed until section 2.3. I will briefly provide a justification for the specific parametrization and, if possible, relate it to other studies.

Starting with β, the parameter controlling the speed of decay in exploration, note that the baseline in this study is about 4 times higher than in Calvano et al. (2020).[17] Conse-

[17] Unfortunately, feature extraction methods require more computational power than tabular learning and the scope of this study does not allow for a lower value of β without sacrificing robustness through the execution of multiple runs per parametrization.

quently, the agents have less time to explore actions and learn from the obtained rewards. It is conjectured that this is partially offset by the introduction of eligibility traces. Also, the variation $\beta = 1 * 10^{-5}$ falls right into the considered parameter grid in Calvano et al. (2020).

The discount factor γ is the only parameter with an inherent economic interpretation. It γ quantifies the time preference of profits today over profits tomorrow. The baseline value 0.95 is chosen to stay consistent with other simulations similar to this study (e.g. Calvano et al. 2020, Klein 2021 and Hettich 2021). In reinforcement learning, the usage of discounting is often void of any theoretical justification. Rather it is commonly employed as a practical mean to avoid infinite reward sums in continuous learning tasks (Schwartz 1993). However, there are some theoretical arguments questioning the validity of discounting in continuous learning tasks with function approximation. In section 4.3, I will discuss results of an alternative setting that attempts to optimize *average rewards*.

Regarding λ, recall that it appears in (14) to update the eligibility trace $\boldsymbol{z}$. In particular, λ controls the *decay rate* of $\boldsymbol{z}$. To understand its purpose, consider first the special case of $\lambda = 0$. This reduces the trace update to $\boldsymbol{z}_t = \frac{\Delta \hat{q}}{\Delta \boldsymbol{w}_t} = \boldsymbol{x}_t$ which is paramount to not using eligibility traces at all.[18] Increasing λ boosts the degree of hindsight by enlarging the algorithm's memory on which parameters were responsible for past action selections. In turn, those evocative parameters are adjusted beyond a one-period horizon. However, increasing λ comes at the cost of high variance. Persistent traces blow up the number of variables in $\boldsymbol{z}$ such that any outcome is imputed to *too* many parameters that have been activated in the distant past.[19] Empirical results show that intermediate values of λ tend to perform best (see e.g. Sutton 1988, Rummery and Niranjan 1994 and Sutton and Barto 2018). Accordingly, I choose a baseline value of $\lambda = 0.5$. The variations comprise experiments with λ between 0 and 0.9.

Parameter	Baseline Value	Variations
β	$4 * 10^{-5}$	$\{1 * 10^{-5},\ 2 * 10^{-5},\ 8 * 10^{-5},\ 1.6 * 10^{-4}\}$
γ	0.95	$\{0,\ 0.25,\ 0.5,\ 0.75,\ 0.8,\ 0.85,\ 0.9,\ 0.99\}$
λ	0.5	$\{0,\ 0.2,\ 0.4,\ 0.6,\ 0.8,\ 0.9\}$
ζ	1	$\{0.1,\ 0.5,\ 1.5\}$
m	19	$\{10,\ 39,\ 63\}$

Table 1: Baseline parametrization and variations.

[18] In fact, the system of (13)-(15) could then be conveniently reduced to a single equation:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha(r_t + \gamma \bar{V}_t(S_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t))\boldsymbol{x}_t$$

[19] The other extreme, $\lambda = 1$, mimics Monte-Carlo algorithms where learning only takes place after an entire *sequence* of time steps concludes. This is unnatural in the context of this study. Accordingly, $\lambda = 1$ is not considered a viable specification.

The parameters ζ and m both control which prices are feasible. The baseline $\zeta = 1$ ensures a symmetric excess of feasible prices above p_m and below p_n. The default value of m is 19.[20] The deliberate choice to depart from the scheme in Calvano et al. (2020) was made to challenge the algorithms with a less prefabricated environment. It comes at the cost of impeded comparability.

2.3. Feature extraction

This section lays out the methods used in this study to map state-action combinations to a set of numerical values $\boldsymbol{x}$. I shall refer to them as *feature extraction methods* (FEM).[21] As outlined in section 2.2.2, the state-action space contains just 3 variables ($p_{i,t-1}$, $p_{j,t-1}$ and $p_{i,t}$). To illustrate FEMs, consider a naive attempt of feature extraction where each S_t and A_t is converted to a feature without mathematical transformation, i.e. $x_1 = p_{i,t-1}$, $x_2 = p_{j,t-1}$ and $x_3 = p_{i,t}$. Every feature is assigned a single coefficient w_d and the estimated value of any state-action combination would then be $\hat{q}(S_t, A_t, \boldsymbol{w}) = \sum_{d=1}^{3} w_d x_d$. Obviously, this simplistic method does not do justice to the complexity of the optimization problem. In particular, a *reward-punishment* theme requires that actions are chosen conditional on past prices (i.e. the state space). Hence, it is imperative to consider interactions between states and actions as well as non-linearities. Table 2 provides an overview of the 4 methods used in this study. I will illustrate and discuss tabular learning as a special case of a linear function here and present the other linear approximation methods in Appendix A.

2.3.1. Tabular learning

A natural way to represent the state-action space is to preserve a distinct feature (and coefficient) for every unique state-action combination. Features are binary, i.e. any feature

Feature Extraction	Parametrization	Length $\boldsymbol{x}$ with $m = 19$	Factor when doubling m
Tabular	-	$m^3 = 6,859$	x8
Tile coding	$T = 5, \psi = 9$	$T\ (\psi - 1)^3 = 2,560$	x1
Polynomial tiles	$T = 5, \psi = 5, k = 4$	$T\ (\psi - 1)^3(\binom{k+3}{3} - 1) = 10880$	x1
Sep. polynomials	$k = 5$	$m(\ \binom{k+2}{2}\ -1) = 380$	x2

Table 2: Feature extraction methods. The third column lists the number of elements in the parameter vector $\boldsymbol{w}$ when $m = 19$. The last column displays the factor by which that length is multiplied if m is doubled.

[20] As indicated earlier, both p_m and p_n can not be played exactly. The variations of m were chosen specifically to encompass feasible prices close to both benchmarks. For instance, $m = 19$ entails one pricing option at 1.466 (close to $p_n = 1.473$) and another at 1.932 (close to $p_m = 1.925$)

[21] The term *feature* is borrowed from the computer science literature. It usually refers to a (transformed) input variable. It is common that the number of features dwarfs the number of original inputs.

is 1 if the associated state-action combination is selected and 0 otherwise:

$$x_d^{Tabular} = \begin{cases} 1 & \text{if } \{p_{1,t-1}, p_{2,t-1}, p_{1,t}\} \text{ corresponds to cell}_d \\ 0 & \text{if } \{p_{1,t-1}, p_{2,t-1}, p_{1,t}\} \text{ does not correspond to cell}_d \end{cases} \tag{19}$$

The respective coefficient tracks the performance over time and directly represents the estimated value of that state-action combination. Accordingly, the length of $\boldsymbol{x}$ is m^3. The approach is called *tabular* because it is easy to imagine a table where every cell represents a unique state-action combination. Tabular methods have been used extensively in the simulations on algorithmic collusion that provided experimental evidence of collusive outcomes being possible in simple environments (see section 1). Their widespread application in repeated pricing games is justified by their conceptual simplicity and their historic usage in autonomous pricing of airline fares and in electricity markets (Ittoo and Petit 2017). Moreover, tabular methods give rise to a family of robust learning algorithms with well-understood convergence guarantees (Jaakkola, Jordan, and Singh 1994).

However, tabular methods are not necessarily the best or fastest way to learn an optimal policy. In real life markets, a salient factor may impede its effectiveness. Prices are continuous - a feature completely disregarded by tabular learning. Ignoring the continuous nature of prices gives rise to two major complications in reality. First, the leeway of decision makers is artificially restricted. Second, due to a *curse of dimensionality*, learning speed and success may deteriorate disproportionately with m. I will take a closer look at both points.

Obviously, any decision maker is only restricted by the currency's smallest feasible increment and can charge more than just a couple of prices. It is certainly conceivable, maybe even desirable, that a decision maker reduces the complexity of a decision process by reducing the number of considered options. However, in most cases it will be impossible to impose such a restriction on competitors. As an extreme example, consider an opponent who never charges the same price twice. Whenever this opponent introduces a new price, a tabular learning agent is coerced to create a new cell in the state-action matrix that will never be revisited. Consequently, the agent continuously encounters new situations from which it can learn, but it is unable to ever utilize the acquired knowledge.

More importantly, tabular learning falls victim to the *curse of dimensionality* and does not scale well with m and n. In the baseline specification of this study, the number of features is $19^3 = 6859$. Doubling m from 19 to 38 causes an eightfold increase of that number to 54872. Even worse, increasing the number of competitors alters the exponent. Changing n from 2 to 3 entails an increase of features by the factor m, in the baseline

specification from 6859 to 130321.[22] It is easy to see that modest increases in complexity have the potential to evoke a disproportionate reduction in learning speed. Indeed, Calvano et al. (2020) show that increasing m and n tends to reduce profits of tabular learning agents.

Another way of looking at the same issue is to consider the nature of prices. They are continuous and transforming them into a qualitative set of discrete actions disregards that fact. In particular, it prevents the opportunity to learn from the result of charging a particular price about the quality of *similar* prices in the same situation. To illustrate with an inflated example, consider a manager who observes large profits after charging a price of 1000. A human manager is able to infer that charging 1001 instead would have yielded a similar profit. Tabular learning agents are not.

The function approximation methods considered in this study alleviate the *curse of dimensionality*. In fact, the length of the feature vector $\boldsymbol{x}$ in *tile coding* and *polynomial tiles* is unaffected by m. For *separate polynomials*, it is proportional to m, i.e. doubling the number of feasible prices also doubles the number of features. Moreover, all methods augment learning in the sense that a particular state-action combination tends to evoke ampler parameter updates that also change the future evaluation of *similar* state-action combinations. Refer to Appendix 2.3 for a detailed description.

2.3.2. Learning speed

I have left open the parametrization of α until now. The reason is that choosing learning speed is not trivial and should depend on which FEM is used. Note that I don't attempt to optimize α, but consider it important to use *reasonable* values to draw valid inference. Principally, every value $\alpha \in [0, 1]$ is possible but values too high place too much emphasis on recent steps while values too low decelerate learning. A natural starting point is to consult other studies on the subject. Calvano et al. (2020) successfully trialed values between 0.025 and 0.25 in the environment presented in section 2.1. In a sequential pricing environment, Klein (2021) shows that performance decreases with values greater than 0.5. These studies provide helpful guidance for tabular learning. However, the range of reasonable values is substantially lower when estimating values with function approximation for two reasons. First, the utilized FEMs update many parameters simultaneously. The advantage of function approximation methods is that they can improve learning speed by inferring the value of many actions from a reward observed due to a single action.

[22] A similar problem arises when the algorithm is supposed to account for cost and demand factors. Every added input, whether due to an additional opponent or any other profit-related variable, increases the number of table cells by a factor of m. While changes in costs and prices are not considered in this study, they obviously play an important role in reality.

Naturally, this comes at the cost of precision and warrants careful adjustments of the parameters to avoid premature valuation techniques. Second, both polynomial FEMs are not binary. In fact, with a high degree k, elements in the feature vector $\boldsymbol{x}$ can become very high. This increases the danger for individual elements to *overshoot* early.

In light of these considerations, I run experiments with various α. Table 3 displays the trialed values. The range for tabular learning is smallest. I explore values down to 10^{-12} for the function approximation FEMs. For the polynomial approaches, I don't even consider values higher than 0.001.

2.4. Convergence considerations

Many reinforcement learning algorithms come with convergence guarantees.[23] To my knowledge, there exist none for the simulations in this study. There are two inhibitors. First, the combination of function approximation, off-policy learning and bootstrapping (i.e. estimating the value of future states) is known to pose a threat of instability (Sutton and Barto 2018). Second, the environment is non-stationary because both agents dynamically change their strategies over time. This engenders a *moving target* problem. The optimal strategy might change over time depending on the other player's learning process (Tuyls and Weiss 2012). Notwithstanding the absence of a guarantee, convergence is nevertheless possible.

3. Results

This section reports on the simulation outcomes of the baseline specification. It is helpful to create a common vocabulary first. I will call every unique combination of FEM and parameters an *experiment*. Every experiment consists of 48 *runs*, i.e. repeated simulations with the exact same set of starting conditions. Lastly, within the scope of a particular *run*, time steps are called *periods*.[24]

FEM	Variations of α
Tabular	$\{0.1, 0.01, 0.001, 10^{-4}, 10^{-5}\}$
Tile coding	$\{0.1, 0.01, 0.001, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 1 * 10^{-8}, 10^{-10}, 10^{-12}\}$
Polynomial tiles	$\{0.001, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-10}, 10^{-12}\}$
Sep. polynomials	$\{0.001, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-10}, 10^{-12}\}$

Table 3: Grid of trialed α by FEM.

[23] Jaakkola, Jordan, and Singh (1994) prove that Q-Learning is guaranteed to converge to an optimal strategy under mild conditions in *stationary* environments. Tsitsiklis and Van Roy (1997) discuss convergence with linear function approximation.

[24] The simulations are run in *R*. The code is available on github (`https://github.com/MalteJe/ai_collusion`). Another technical note: The program seed does not vary between experiments, i.e. for every

3.1. Convergence

Notwithstanding the lack of a theoretical convergence guarantee (see section 2.4), prior experiments have shown that simulation runs tend to approach a stable equilibrium in practice (see e.g. the studies mentioned in section 1). Note that I use a descriptive notion of equilibrium that is detached from economic theory. It is characterized by the simple observations that the same set of prices continuously recur over a longer time interval.

The following, arbitrary, but practical convergence rule was employed. If a price cycle recurred for 10,000 consecutive periods, the algorithm is considered *converged* and the simulation concludes. Both agents' adherence is required.[25] For efficiency reasons, price cycles up to a length of 10 are considered and a check for convergence is only undertaken every 2,000 periods. If no convergence is achieved until 500,000 periods, the simulation terminates and the run is deemed *not converged*. Furthermore, there are a number of runs that *failed to complete* as a consequence of the program running into an error. Unfortunately, the program code does not allow to examine the exact cause of such occurrences in retrospect. However, the failed runs only occurred with unsuitable specifications (see below).

In accordance with the outlined convergence criteria above, Figure 1 displays the share of runs that, respectively, converged successfully, did not converge until the end of the simulation or failed to complete. Two main conclusions emerge. First, failed runs are only prevalent in the polynomial tile experiment with $\alpha = 0.001$. Second, the tiling methods are more likely to converge. Both points deserve some further elucidation.

Regarding the failed runs, recall from section 2.3.2 that features of polynomial extraction are not binary and warrant cautious adjustments of the coefficient vector. I suspect that with unreasonably large values of α, the estimates of $\boldsymbol{w}$ overshoot early in the simulation, diverge and at some point exceed the software's numerical limits.[26] While important to acknowledge, the failed runs are largely an artifact of an unreasonable specification and I will not account for them for the remainder of this text.

Out of the completed runs without program failure, 95.8% did converge. Interestingly, there are subtle differences between FEMs. With only one exception, both tiling methods converged consistently for various α. With only 89.3% of runs converging, separate poly-

run there is a *sibling run* with an equivalent initialization of the random number generator in every other experiment. The seed determines the initial state and the decision in which periods the agents decide to explore.

[25] Of course it is possible that the cycle length differs between agents. For instance, one agent may continuously play the same price while the opponent keeps alternating between two prices. In this case, the cycle length is 2.

[26] Controlled runs where I could carefully monitor the development of the coefficient vector $\boldsymbol{w}$ seem to confirm the hypothesis.

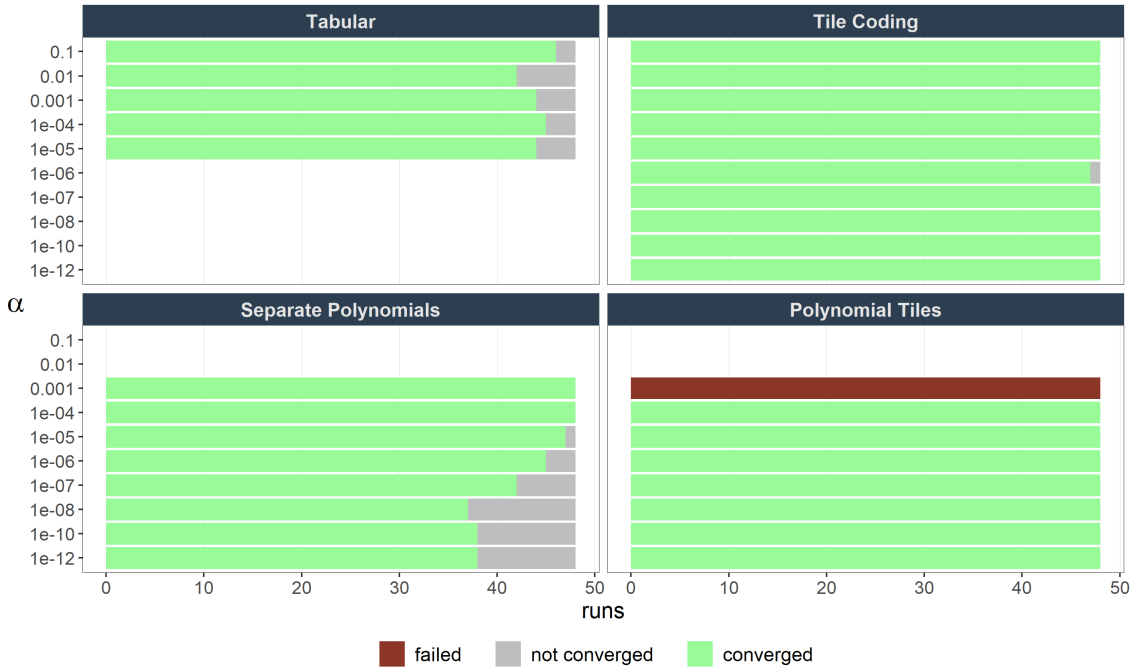

Figure 1: Number of runs per experiments that (i) achieved convergence, (ii) did not converge or (iii) failed to complete as a function of FEM and α.

nomials constitute the other extreme. The figure also indicates that convergence becomes less likely for low values of α. With tabular learning, 92.1% of runs converged without clear relation to different values of α.

Figure 2 displays a frequency polygon of the runs that achieved convergence within 500,000 periods. Clearly, the distribution is fairly uniform across FEMs. Most runs converged between 200,000 and 300,000 runs. This is an artifact of the decay in exploration as dictated by β. Before the focal point of 200,000 is reached, agents probabilistically experiment too frequently to observe 10,000 consecutive periods without any deviation from the learned strategies. Thereafter, it becomes increasingly likely that both agents keep *exploiting* their current knowledge and continuously play the same strategy for a sufficiently long time to trigger the convergence criteria. Note that the low quantity of runs converging between 300,000 and 500,000 suggests that increasing the maximum of allowed periods would not necessarily produce a significantly higher share of converged runs. Appendix B.1 unveils that the choice of the FEM affects cycle length and the range of prices agents charge upon convergence. Next, I proceed by examining profits.

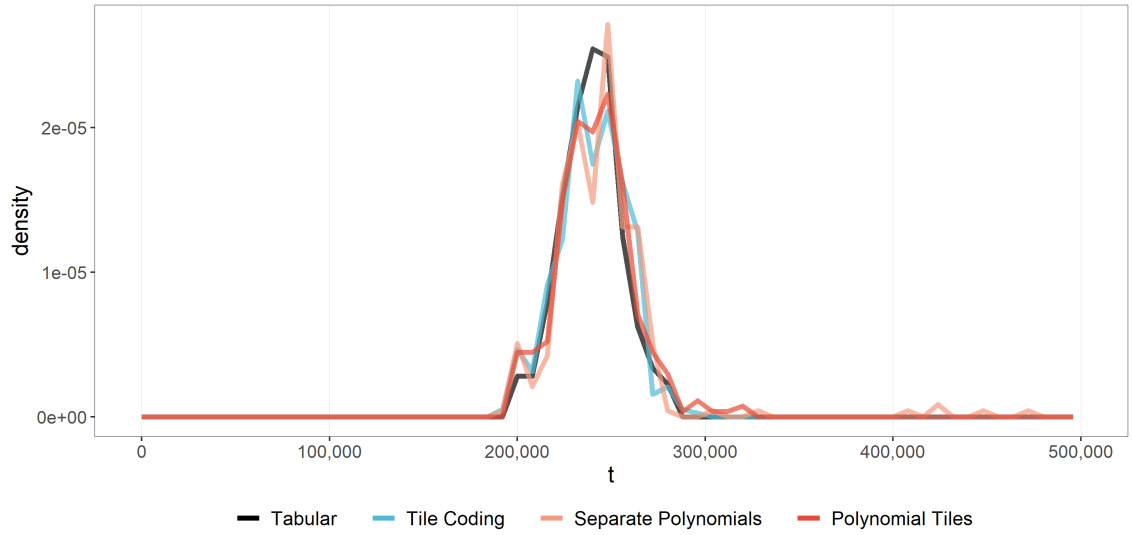

Figure 2: Timing of convergence by FEM. Only includes converged runs. Width of bins: 8,000.

3.2. Profits

In order to benchmark the simulation profits, I normalize profits as in Calvano et al. (2020) and Hettich (2021):

$$\Delta = \frac{\bar{\pi} - \pi_n}{\pi_m - \pi_n} \quad , \tag{20}$$

where $\bar{\pi}$ represents profits averaged over the final 100 time steps upon convergence and over both agents in a single run.[27] The normalization implies that $\Delta = 0$ and $\Delta = 1$ respectively reference the Nash and monopoly solution. I will denote these special cases as Δ_n and Δ_m. Note that it is possible to obtain a Δ below 0 (e.g. if both agents charge prices equal to marginal costs), but not above 1.

Figure 3 displays the convergence profits as a function of FEM and α.[28] Every data point represents one experiment, more specifically the mean of Δ across all runs making up the experiment. First of all, note that average profits consistently remain between both benchmarks p_m and p_n across specifications.[29] As with prior results, the plot unveils salient differences between FEMs. On average, polynomial tiles runs yield the highest profits. The average Δ peaks at 0.84 for $\alpha = 10^{-8}$. Higher values of α tend to progressively

[27] Naturally, in the case of non-converged runs I use the final 100 episodes before termination.

[28] For comparison purposes, I also show the closest available experiment from Calvano et al. (2020) ($\beta = 2\ x\ 10^{-5}$, $m = 15$). The disparity to tabular learning runs in this study can be explained by differences in parameter choices and experiment setup (e.g. initialization of Q-Matrix).

[29] There are three exceptions that are hidden in the plot to preserve reasonable y axis limits. More specifically, for the FEM polynomial tiles with $\alpha = 0.0001$, the average Δ is -1.72. With separated polynomials, the average Δ with, respectively, $\alpha = 0.0001$ and $\alpha = 0.001$ is -1.86 and -0.282. This refines the observation from section 3.1. It appears that high values of α converge in equilibrium strategies void of any reasonableness. In the case of polynomial tiles, the program even crashes due to diverging parameters.

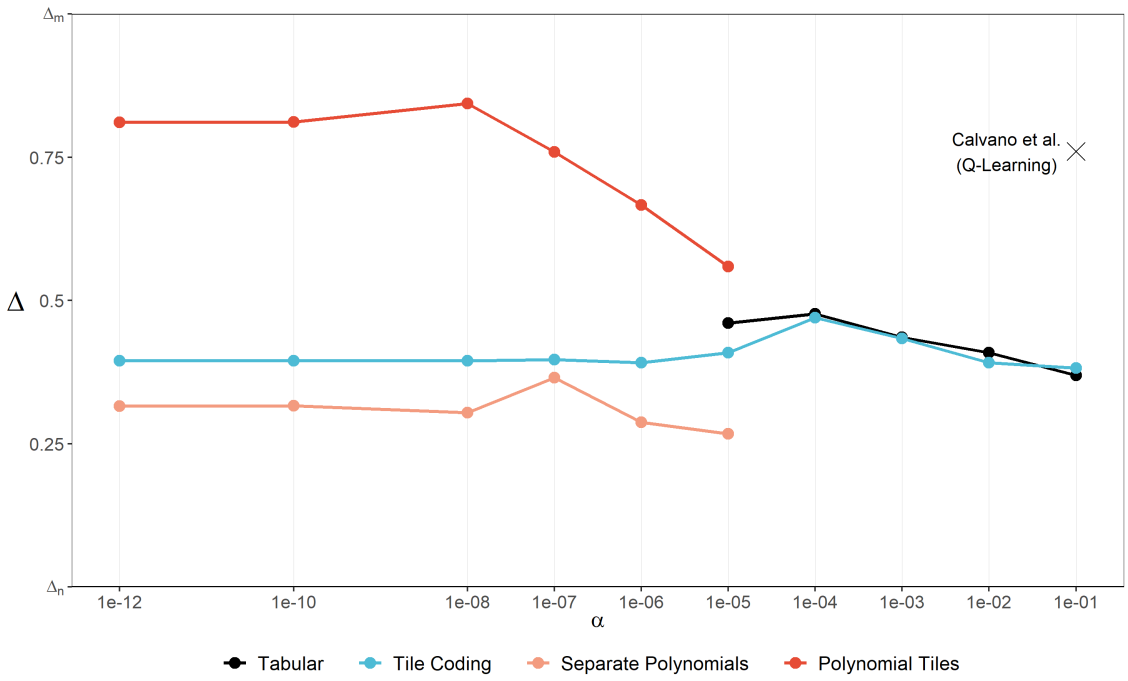

Figure 3: Average Δ by FEM and α. Includes converged and non-converged runs. Three experiment (polynomial tiles with $\alpha = 0.0001$ and separated polynomials with $\alpha \in \{0.001, 0.0001\}$) are excluded for better presentability. An exemplary experiment from Calvano et al. (2020) is provided for comparison purposes. Note the logarithmic x-scale.

decrease profits. Moving downwards on the y-axis, both tabular learning and tile coding yield similar average values of Δ. Furthermore, the level of α does not seem to impact Δ much. For both methods $\alpha = 10^{-4}$ induces the highest average Δ at 0.48 and 0.47 respectively. Similarly for separate polynomials, Δ does not seem to respond to variations in α. The maximum Δ is 0.37.[30]

These insights establish that, what constitutes a sensible value of α clearly depends on the selected FEM. Therefore, for the remainder of this text, I will select an optimal α for every FEM and present further results only for these combinations. In determining *optimality* of α, I do not rely on a single hard criteria. Instead, I consider a number of factors including the percentage of converged runs, comparability with previous studies and prefer to select experiments with high average Δ as they are most central to the purpose of this study. Table 4 provides a justification for every experiments setting deemed *optimal*. To get a sense of the variability of runs within the optimized experiments and the price trajectory over time, Appendix B.3 contains further visualizations of the development of prices and profits of all runs with optimized α.

[30]Naturally, averaging Δ over all runs of an experiment, as done to create Figure 3, has the potential to hide subtleties in the distribution of Δ. Appendix B.2 shows that Δ varies quite a bit between runs of an experiment but the main points of this section remain valid.

FEM	α	Justification
Tabular	0.1	- comparability with previous simulation studies - most pronounced response to price deviations (see section 3.3)
Tile Coding	0.001	- high Δ - most pronounced response to price deviations (see section 3.3)
Separate Polynomials	10^{-6}	- high percentage of converged runs
Polynomial Tiles	10^{-8}	- high Δ

Table 4: *Optimized* values of α by FEM

3.3. Deviations

This section examines whether the learned strategies are stable in the face of deviations from the learned behavior. There are at least two explanations for the existence of supra-competitive outcomes. First, agents simply fail to learn how to compete effectively and miss out on opportunities to undercut their opponent. Second, agents avoid deviating from the stable strategy because they fear retaliation and lower (discounted) profits in the long run. Importantly, only the latter cause, supra-competitive prices underpinned by some form of a *reward-punishment scheme*, allows to label the outcomes as *collusive* and warrants attention from competition policy (Assad et al. 2020). Therefore, I conducted an artificial *deviation experiment* to scrutinize whether agents learn to actually retaliate in the wake of a deviation. The bottom line of that exercise is that only tabular learning evokes a conspicuous punishment from the non deviating agent. Before the results are discussed in detail, I follow with a short portrayal of the deviation experiment.

Denote the period in which convergence was detected as $\tau = 0$. At this point, both agents played for 10,000 periods an equilibrium strategy they mutually regard as optimal. At $\tau = 1$, I force one agent to deviate from her learned strategy and play instead the short-term best response that mathematically maximizes immediate profits. Subsequently, she reverts to the learned strategy. In order to verify whether the non deviating agent proceeds to punish the cheater, he sticks to his learned behavior throughout the deviation experiment. In total, the deviation episode lasts 10 periods. Learning and exploration are disabled (i.e. $\alpha = \epsilon = 0$). In order to evaluate the deviation, it appears useful to define a *counterfactual* situation where both agents stick to their learned strategies for another 10 periods. Comparing (discounted) profits between the experiment and the counterfactual allows to assess the profitability of the deviation.

As the responses to one agent's deviation vastly differ across FEMs, it is natural to discuss them separately at first and contrast differences only thereafter. It is difficult to summarize all information in a single graph or table, so I will consult Figure 4, Figure 5

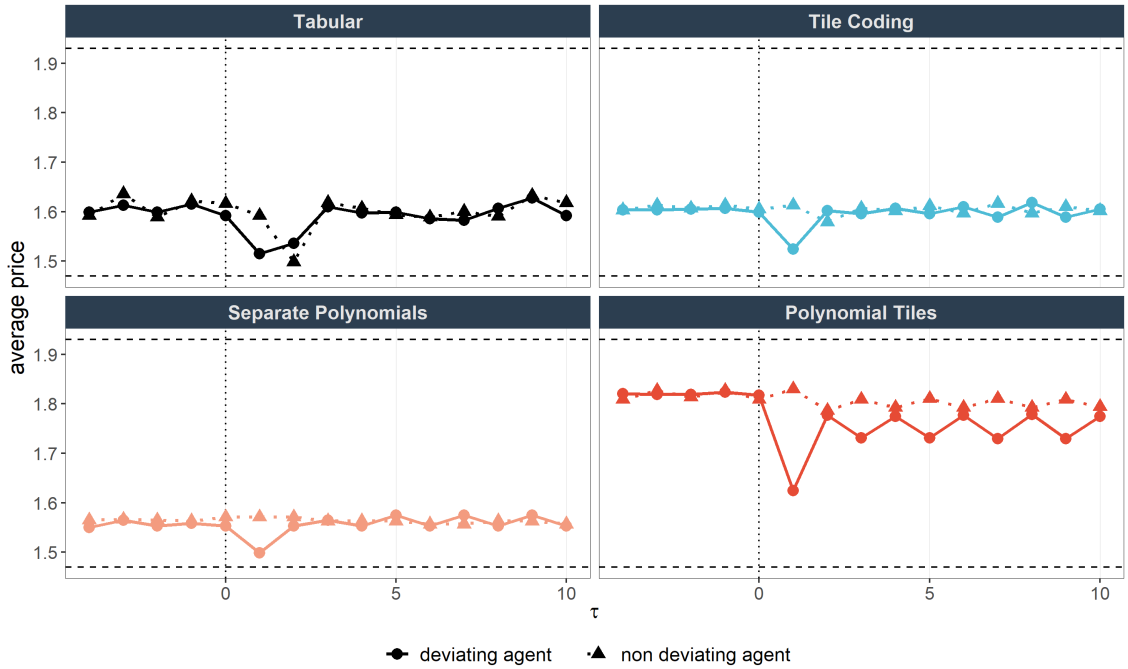

Figure 4: Average price trajectory around deviation by FEM. Points represent the average price over all runs of an experiment. Dashed horizontal lines represent the fully collusive price p_m and the static Nash solution p_n. Dotted vertical line reflects time of convergence, i.e. the period immediately before the forced deviation.

and Table 5 simultaneously to describe the deviation and response patterns. Before that, a brief description is in line. Figure 4 displays the price trajectory around the forced deviations averaged over all runs of an experiment. Since the average price trajectory might veil important differences between runs, Figure 5 illustrates the range of deviation and punishment prices compared to the counterfactual price that would have materialized if no deviation had taken place and agents kept following their learned strategies. Note that in the presence of price cycles, part of the variation can be explained by *cycle shifting*, a phenomenon where the agents return to the learned cycle but the intervals are not aligned with the counterfactual path. These differences should even out over all runs of an experiment and therefore, not systematically bias the boxes in either direction. Similarly, the average price response in Figure 4 is largely unaffected by this phenomenon. Finally, Table 5 reports the share of deviations that turned out to be profitable compared to the counterfactual.[31]

As indicated earlier, the non-deviating agents in tabular learning runs learned to punish deviations. Figure 4 shows that the non deviating agent, on average, undercuts the deviation price at $\tau = 2$. Simultaneously, the deviating agent already begins reverting to pre-deviation prices. However, this result's general validity is qualified. Figure 5 unveils

[31] Appendix B.4 contains further visualizations of the deviation experiments.

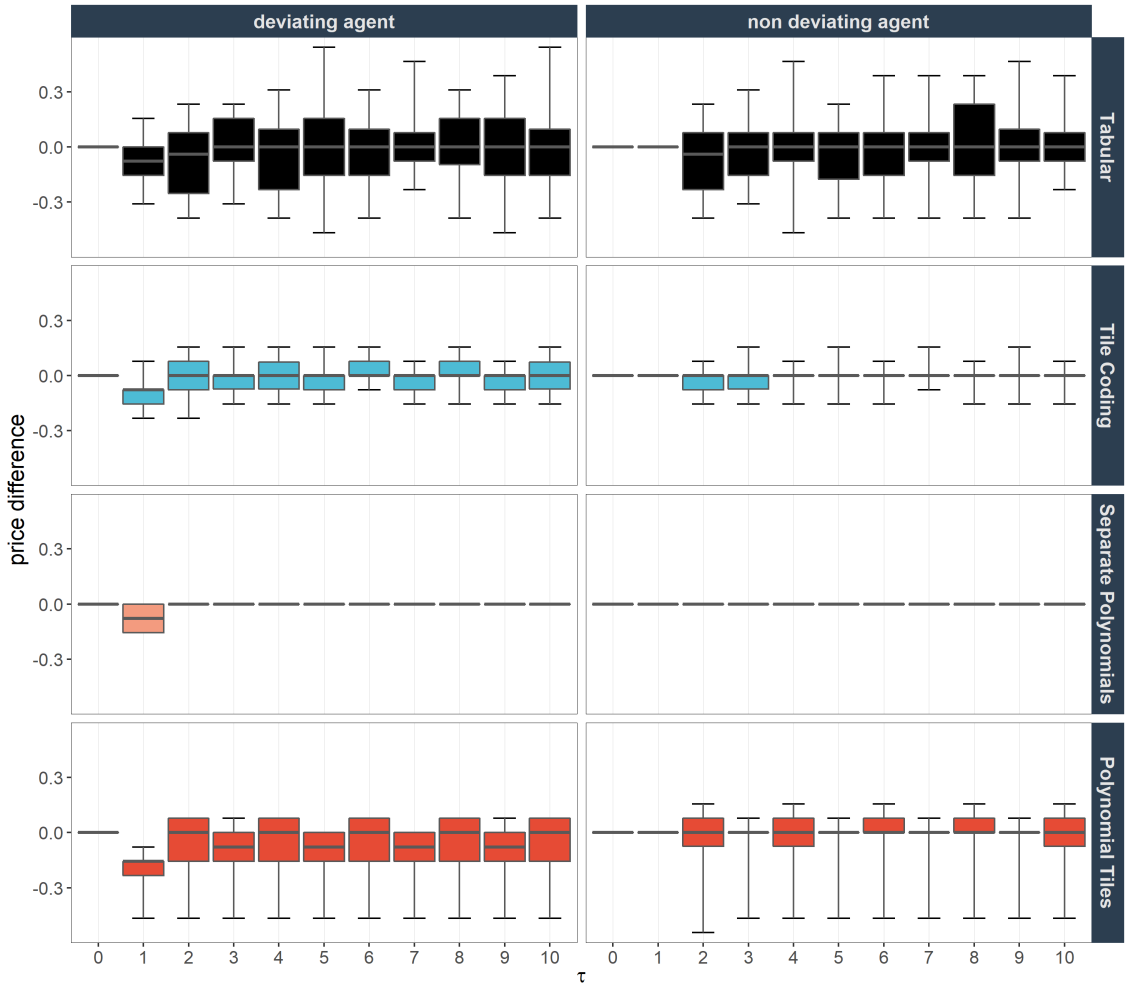

Figure 5: Distribution of price differences around deviation relative to counterfactual path *without* forced deviation, i.e. the difference to the price had no deviation taken place, by FEM. Only includes converged runs because a clear counterfactual exists. Boxes demarcate 15th and 85th percentiles. They are extended by whiskers that mark the entire range of price differences. Horizontal lines represent the group median.

that the non deviating agent does not always reduce prices compared to the counterfactual. Despite the existence of punishment prices in some runs, agents are fairly quick to return to the price levels observed before the deviation was forced upon them. As early as $\tau = 3$ there is no visible difference between average pre- and post-deviation price levels.[32] This might partly follow from prices being relatively close to the Nash equilibrium in the first place. The punishments ensure that deviating is (strictly) profitable in only 24% of runs. This suggests that, upon convergence, agents stick to a stable equilibrium, from which deviations tend to be unprofitable due to the cheated agent retaliating. These findings confirm the results from previous studies.

When examining the outcomes of the other FEMs, different conclusions emerge. Recall from Figure 3 that tile coding yielded convergence profits very similar to tabular learning. Yet, Figure 4 and Figure 5 only hint at slight punishments in some runs. In fact, the median of the cheated agent's price at $\tau = 2$ is exactly 0, which amounts to a complete absence of a response. This lack of punishment renders 56% of the cheater's deviations profitable. In light of that, it is surprising that the cheating agent tends to return to pre-intervention price levels instead of continuing to exploit her opponent's failure to punish deviations.

This is even more obvious for the separate polynomials FEM. The deviation responses are easy to summarize. After the forced intervention at $\tau = 1$, both agents immediately return to the pre-deviation equilibrium in all runs. This is remarkable for two reasons. First, the non deviating agent completely fails to punish the cheater's behavior and does not respond to the price cut whatsoever. Consequently, 69% of the deviations are profitable.[33] This leads to the second point. Despite the obvious advantage of cheating, the deviating agent returns to the pre-deviation price without exception, thus failing to exploit her opponent's weakness. To put this in the right context, remember that the initial price levels are fairly close to the Nash equilibrium and the deviation's profitability is relatively small compared to the potential gains realizable in other experiments (see also Appendix B.4). Still, it appears puzzling that such a simple strategy improvement remains consistently untapped. A potential explanation is this. The FEM *separate polynomials* preserves a distinct set of parameters for every feasible action a (see Appendix A.4).[34]

[32] Previous studies showcase a strong deviation is usually followed by a more gradual reversion to pre-deviation behavior (around 5-10 periods), see in particular Figure 4 in Calvano et al. (2020) and Figure 3 in Klein (2021).

[33] The remaining 31% comprise runs where the deviating agent was already playing the short-term best response. Remember that the separate polynomials FEM tends to converge with prices at or close to the Nash equilibrium.

[34] As opposed to the two other function approximation FEMs that treat the action space as a continuous variable.

FEM	agent	share profitable	share unprofitable
Tabular	deviating	0.24	0.57
Tabular	non deviating	0.07	0.74
Tile Coding	deviating	0.56	0.31
Tile Coding	non deviating	0.04	0.83
Separate Polynomials	deviating	0.69	0.00
Separate Polynomials	non deviating	0.00	0.69
Polynomial Tiles	deviating	0.85	0.15
Polynomial Tiles	non deviating	0.04	0.96

Table 5: Share of profitable and non-profitable deviations by FEM and agent. Deviations are deemed *profitable* if the discounted profits until $\tau = 10$ due to the deviation exceed profits from a counterfactual without deviation. Only includes converged runs because a clear counterfactual exists. Discounting is equivalent to γ in (13), i.e. 0.95. A significant number of deviations are neither profitable nor unprofitable. In those runs, the learned strategy of the deviating agent is actually the best response at $\tau = 1$ and both agents keep following their respective price cycle.

It appears that the set for the preferred action is constructed in a way that yields high estimated values for that action *irrespective of the state*. In its effect, this is paramount to a simplistic valuation technique involving just *one* parameter per action. With that approach, the agent completely disregards the state set and simply plays the preferred action at all times. A generous interpretation is that this is similar to a static Nash equilibrium. However, many runs converge in price levels way above that. To conclude, the agents' failure to play economically sound strategies casts doubts on the viability of the FEM in reality.

Finally, consider the experiment with polynomial tiles. Recall that this experiment generated prices closest to the monopoly benchmark. Despite that, the deviation experiment for polynomial tiles leads to similar conclusions to the ones with separate polynomials. Though there are some variations between runs that warrant detailed examination. Consider first the non deviating agent. Again, the majority of runs exhibits a failure to respond to the price cut. However, selected runs show a *matching* strategy where the cheated agent meets the price cut with a similar price. Notably, in those circumstances, agents *do not return* to the previously learned path but quickly establish a new equilibrium. Moreover, note that Figures 4 and 5 display a slight bias downwards over all periods. This is indicative of *continued cheating* of the deviating agent. After being forced to undercut the price, she proceeds to set prices below pre-deviation levels without getting punished. This, too, results in a new equilibrium.[35] In light of high pre-deviation prices and the lack of retaliatory prices, it is unsurprising that 85% of deviations are profitable.

[35] Figure 18 in Appendix B illustrates both phenomena (price matching and continued cheating) through the exact price sequence of exemplary runs.

To summarize the deviation experiments, tabular learning agents learned to collude and tend to support the supra-competitive equilibria with a reward-punishment scheme. On the other hands, baring a few exceptions, the non deviating agents in experiments with the FEMs utilizing function approximation *fail to respond to price cuts* and are easy to exploit. Moreover, the deviating agents tend to leave that weakness unexploited. Overall the deviation exercise suggests that while algorithmic agents manage to sustain high prices when playing each other, their strategies are incomplete and easy to exploit.

Evidently, under the regime of this study's simulations, tabular learning is better in producing stable supra-competitive outcomes than the function approximation FEMs. To illustrate the notion of *stability* in this context, consider the following thought experiment of a *superagent*. Upon convergence, a rational player with perfect information about the economic environment and the learned policies of both agents enters the game and takes over pricing authority from one of them. Importantly, the superagent can anticipate the opponent's price in the next period and calculate the short-term maximizing response as well as the opponent's reaction to the deviation and so on. When playing against a tabular learning agent, the superagent would deliberately stick to the convergence pricing scheme as cheating is sure to evoke a retaliation rendering a deviation unprofitable (see Table 5). Contrary, when facing an opponent who learned its strategy through a function approximation FEM, the superagent could easily cheat on the opponent to increase short-term profits without being punished in subsequent periods.

The evidence also suggests that function approximation creates hesitation in the agents to change best responses. Probabilistically, *exploration* ensures that both agents will undercut the price of their opponent and realize excess profits similar to those in the forced deviation experiment. However, it appears that agents fail to learn (enough) from such *explored cheating*. As evidenced by the undertaken deviation experiment, they typically return to the pre-deviation price (cycle) immediately. This rigidity in adjusting strategies potentially points to a problem with the specific algorithm or the tuning of its parameters. For instance, a higher α could enable the cheater to learn faster that an unpunished deviation is more profitable than adhering to the learned strategy.

Recall that learning and exploration were turned off for the deviation experiment. This gives rise to an objection to the presented results. The non deviating agent, stripped of the ability to adjust his strategy, might only be exploitable for a finite number of periods until he adjusts his strategy. In fact, a generosity to condone isolated price cuts might be conducive to establishing high price levels early in the simulation runs. However, Appendix B.5 demonstrates that the lack of punishment in response to a deviation remains

ubiquitous in prolonged deviation experiments with enabled learning ($\alpha > 0$).

4. Robustness and variations

To show that the presented findings are not an artifact of specific experiment design choices, this section performs a variety of robustness checks and reports results on slightly altered learning schemes. In all additional experiments I hold α fixed at its optimal values (see Table 4). I will show that the considered variations do not impact the simulation results much. Rather, they reinforce the conclusions from the last section. In sections 4.1 and 4.2 I vary the parameters controlling the learning process and the price grid that is available to both agents. I consider an alternative reward setting in section 4.3.[36]

4.1. Learning parameters

Besides the learning rate α, the exploration strategy is arguably the most important steering choice in reinforcement learning. As discussed, β controls the decay in exploration over time. To assess its impact on the sensitivity of outcomes, I run a number of experiments varying β while keeping the manually optimized values of α constant (see Table 4).[37] Naturally, I proportionately adjust the number of maximal periods before a run is forced to terminate.[38]

Figure 6 displays that the impact of exploration on average Δ is relatively small across FEMs. Interestingly, applying the deviation routine described in section 3.3 uncovers that extended exploration supports the stability of the convergence equilibrium only in the case of tabular learning. Table 6 clearly shows that cheating becomes less profitable when the non-deviating agent utilizing tabular learning had more opportunities to explore reactions to a deviation. The share of profitable deviations ranges from 39% at $\beta = 0.00016$ to 6% at $\beta = 2 * 10^{-5}$. Strangely, the separate polynomials FEM shows the opposite pattern. Less exploration makes deviations less attractive. My interpretation is that the overall lower prices levels associated with less exploration indicate that the deviating agent already plays the best response in many runs. Lastly, the value of β does not seem to have a large impact on either of the tiling FEMs. Their convergence equilibria are unstable for

[36] I also consider variations to the discount factor in Appendix C.5. Appendix C.6 shows that the findings are also robust to amended versions of Algorithm 1.

[37] Note that these values are not necessarily optimized for alternative β. Ideally, exploration rate and learning speed should not be considered in isolation. Indeed, Calvano et al. (2020) show that lower values of α perform better if exploration is extensive. However, the scope of this study does not allow to systematically search over a 2-dimensional grid of α and β.

[38] With the lowest and highest values of β (0.00016 and 10^{-5}), the maximum number of periods is adjusted to 125000 and 2000000 respectively.

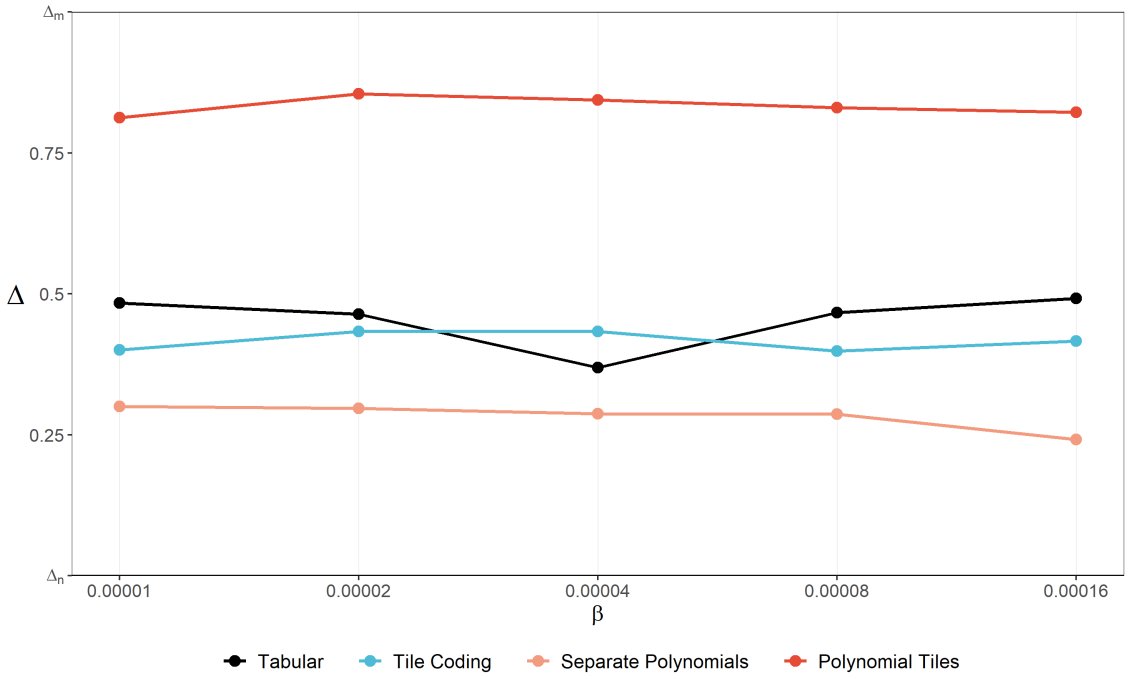

Figure 6: Average Δ by FEM and β. Includes converged and non-converged runs.

all trialed values of β.[39]

4.2. Price grid

Table 2 emphasized that the length of the parameter vector $\boldsymbol{w}$ with tabular learning increases disproportionately with m. Likewise, the optimization problem is likely to become more complex. On the other hand, the feature extraction mechanisms of tile coding and polynomial tiles are largely unaffected by m. Recall the baseline specification with $m = 19$. To gauge the effect on outcomes, I executed experiments with additional variations, specifically $m = 10$, $m = 39$ and $m = 63$. Due to computational restrictions, these experiments

FEM	agent	$\beta = 0.00016$	$\beta =$ 8e-05	$\beta =$ 2e-05	$\beta =$ 1e-05
Tabular	deviating	0.39	0.38	0.06	0.08
Tabular	non deviating	0.20	0.09	0.00	0.04
Tile Coding	deviating	0.50	0.54	0.56	0.67
Tile Coding	non deviating	0.04	0.08	0.04	0.00
Separate Polynomials	deviating	0.58	0.67	0.89	0.90
Separate Polynomials	non deviating	0.00	0.00	0.00	0.00
Polynomial Tiles	deviating	1.00	0.92	0.92	0.91
Polynomial Tiles	non deviating	0.02	0.02	0.02	0.00

Table 6: Share of profitable deviations by FEM, agent and β. Annotations from Table 5 apply.

[39] Appendix C.1 supplements the observation of this section by showing that punishment severity and length also increase with extended exploration. Moreover, in Appendix C.2 I briefly discuss that the choice of λ does have little impact on deviation behavior. But the variance in the distribution of profits between runs seems to increase with high values of λ.

only comprise 16 runs. Accordingly, inference should be treated with care.

Unsurprisingly, convergence becomes less likely when m increases. While all runs with $m = 10$ converged, the percentage for $m = 39$ and $m = 63$ is only 67.2% and 57.8% respectively. Despite that, Figure 7 indicates that varying m does not seem to have much of an impact on the average Δ. On first glance, this seems confusing. As the complexity of the problem increases, one would expect agents to struggle with optimizing their strategy. However, the puzzle is partly solved by taking into account the stability of the equilibrium. Table 7 suggests that the share of profitable deviations increases with m for tabular learning and the polynomial tiles FEM. Most notably, the share of profitable punishments in runs with tabular learning increases from only 12% when $m = 10$ to 67% when $m = 63$. Interestingly, the share of profitable deviation in runs with polynomial tiles is also significantly smaller when $m = 10$. Recall that the FEM evoked a weak punishment and gave rise to new equilibria in *some* runs. This tendency seems to be reinforced when m is low.

In summary, increasing the environment's complexity through the number of available prices m makes supra-competitive outcomes not less likely, but less stable in the face of deviations. Appendix C.3 supports this hypothesis by showing that punishments seem to be strongest with $m = 10$ and that an increase in the share of *viable* prices in the range of p_m and p_n (i.e. a lower ζ) seems to lower convergence profits. Despite these subtleties, the qualitative conclusions from section 3 remain intact.

4.3. Differential reward setting

In reinforcement learning, discounting is commonly used to avoid infinite value accumulation, but rarely has a practical interpretation (Schwartz 1993). Therefore, the blend with an economic task seems natural. However, despite wide usage, Naik et al. (2019) argue that discounting in combination with function approximation is fundamentally incompat-

FEM	agent	m = 10	m = 19	m = 39	m = 63
Tabular	deviating	0.12	0.24	0.42	0.67
Tabular	non deviating	0.00	0.07	0.08	0.33
Tile Coding	deviating	0.69	0.56		
Tile Coding	non deviating	0.00	0.04		
Separate Polynomials	deviating	0.88	0.69	0.93	0.87
Separate Polynomials	non deviating	0.00	0.00	0.00	0.20
Polynomial Tiles	deviating	0.69	0.85	1.00	0.94
Polynomial Tiles	non deviating	0.06	0.04	0.00	0.00

Table 7: Share of profitable deviations by FEM, agent and m. Annotations from Table 5 apply. Empty cells are a consequence of no converged runs for the particular experiment.

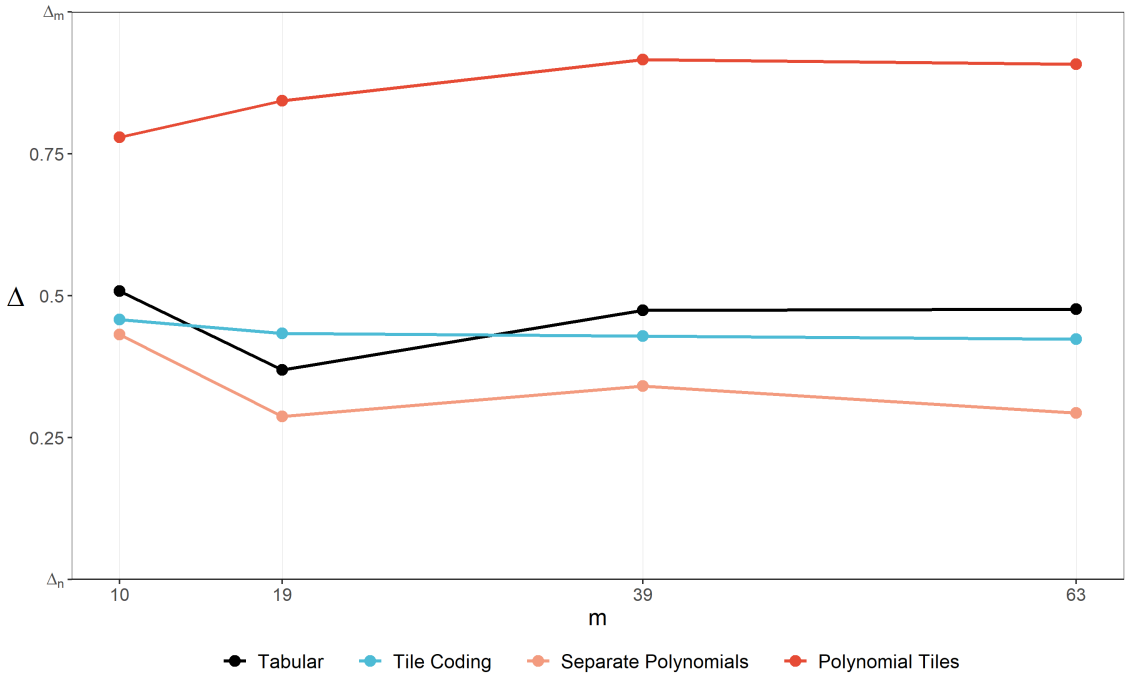

Figure 7: Average Δ by FEM and m. Includes converged and non-converged runs.

ible in infinite sequences. They suggest an alternative *differential reward* setting, where (13) is replaced by:[40]

$$\delta_t^{differential} = R_t - \widetilde{R}_t + \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t) \quad , \tag{21}$$

where $\widetilde{R}_t$ is a (weighted) average reward periodically updated according to

$$\widetilde{R}_{t+1} = \widetilde{R}_t + \upsilon r_t \quad , \tag{22}$$

where υ is a parameter controlling the speed of adjustment. The formulation ensures that recent rewards are weighted higher. The rest of Algorithm 1 remains untouched. Note that the differential reward setting does not involve any discounting. At first glance, this clashes with the economic understanding of time preferences. However, there are two arguments why the differential reward setting might still be well suited. First, Sutton and Barto (2018) proof that, due to the infinite nature of the Bertrand environment, the ordering of policies in the discounted value setting and the setting with average rewards are equivalent (irrespective of γ). Second, pricing algorithms tend to be used in markets with frequent price changes where it is less important whether a profit is realized immediately or in the next period.

I conducted a series of experiments varying over the following values of υ: 0.001, 0.005,

[40] See chapter 10 in Sutton and Barto (2018) for a rigorous treatment. Hettich (2021) shows that the differential reward setting works well with agents in a Bertrand environment.

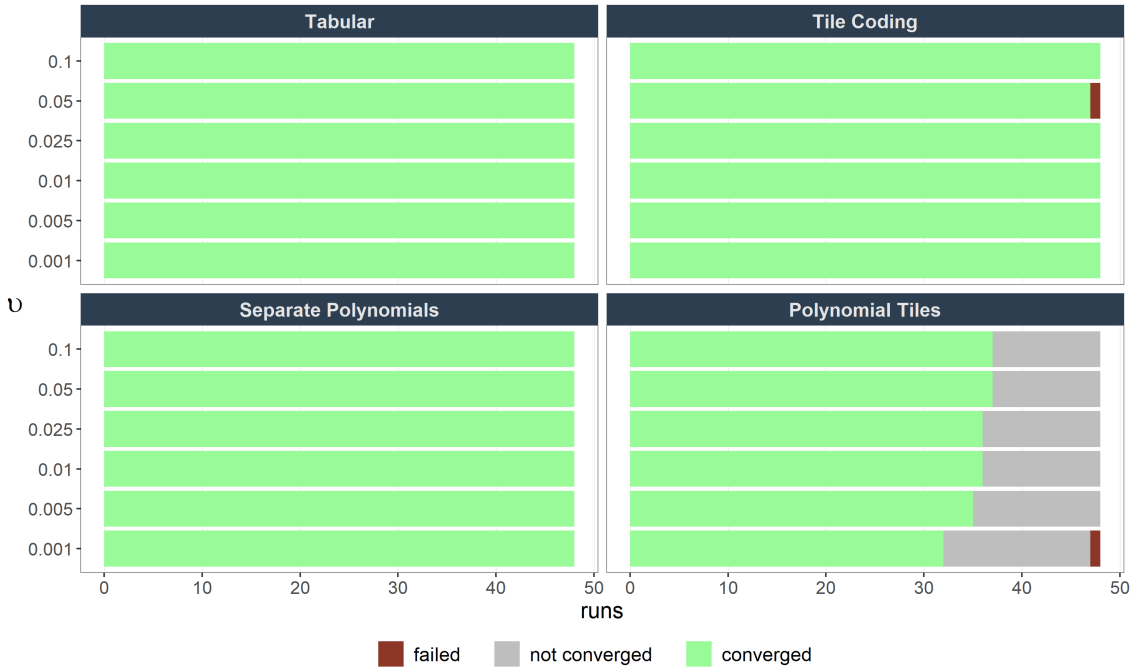

Figure 8: Number of runs per experiments that (i) achieved convergence, (ii) did not converge or (iii) failed to complete by FEM and υ.

0.01, 0.025, 0.05 and 0.1. As with the other variations, α is fixed at values deemed optimal. Figure 8 shows the share of converged runs as a function of υ and the FEM. Disregarding two runs that failed to complete, convergence is consistently achieved for tabular learning, tile coding and separate polynomials. Contrary, only 74.2% of polynomial tiles runs converged. This starkly contrasts the observation made in the experiments using the discounted reward setting. There, all runs with polynomial tiles converged for various values of α. Moreover, the plot suggests that low values of υ impede convergence for this FEM.

Figure 9 displays how the average profits relative to p_n and p_m change with υ. The overall impact is small. However, tabular learning and, to a lesser extent, tile coding seem to converge at higher profits when υ is very low. With respect to punishment of price cuts, the results are similar to the discounted setting. Irrespective of υ, the majority of deviations in experiments with separate polynomials and polynomial tiles is profitable and evokes no retaliation. With tabular learning, the share of profitable deviations is 24.7% over all runs. There are hints of evidence that some sort of punishment in tile coding is more pronounced in the differential reward setting. Only 48.8% of deviations are strictly profitable. Appendix C.4 shows that non deviating agents retaliate with a price cut at $\tau = 2$ in some runs.

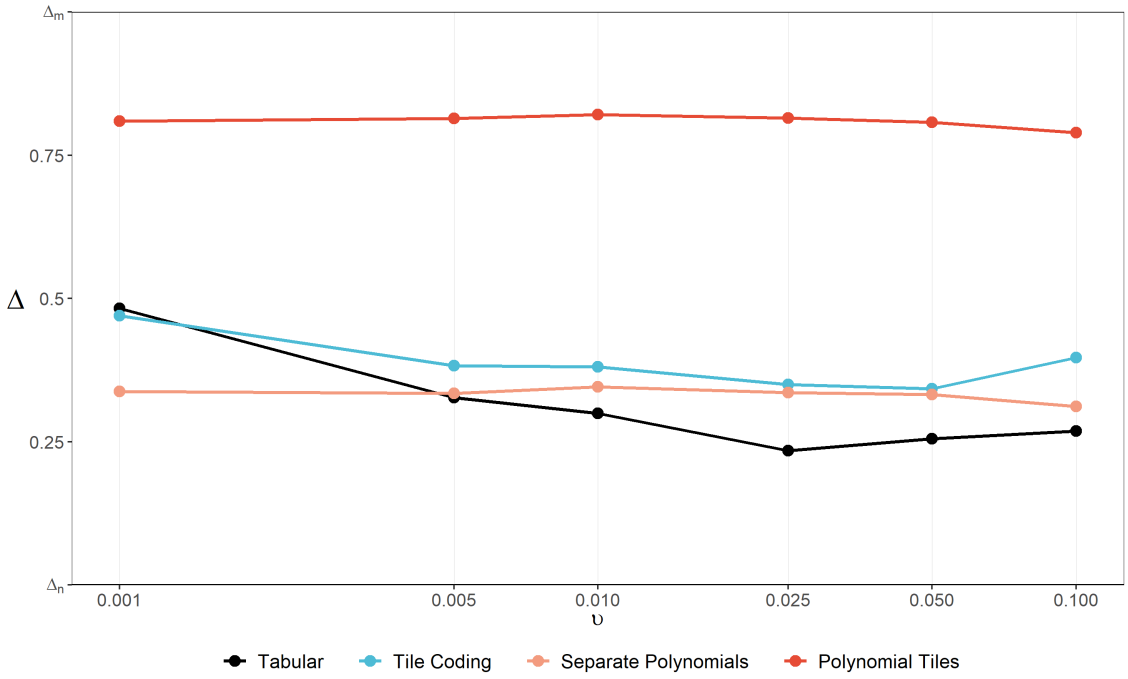

Figure 9: Average Δ by FEM and υ. Includes converged and non-converged runs. Note the logarithmic x-scale.

5. Conclusions

The motivation for this paper is that, despite the success of Q-Learning in achieving collusion in repeated games of price competition, tabular learning methods might face practical challenges when applied to real markets. I developed three linear function approximation methods that scale better with the learning task's complexity and combined them with suitable reinforcement learning algorithms. To assess their merit, I deployed them to a simultaneous pricing environment and compared their performance to tabular learning.

The simulations have shown that all FEMs tend to converge in supra-competitive profits as long as parameter specifications remain reasonable. As is shown in other studies, tabular learning agents acquire truly collusive strategies and show a stubborn resilience to return to the convergence equilibrium after episodes of forced deviation. On the contrary, the convergence equilibra arising in simulations with function approximation FEMs are not supported by a reward-punishment scheme. I show in various deviation exercises that agents have not learned to systematically punish price cuts. Thus, the simulation *failed to provide evidence of collusion with function approximation FEMs*. Furthermore, despite the obvious lack of a credible deterrent, deviating agents are unable to exploit that weakness and return to pre-deviation prices. This is clear evidence of irrational behavior on the side of the agents. These findings also apply when using different algorithms and are robust to variations in learning and environment parameters. In particular, the introduction of

eligibility traces does not qualitatively change the conclusions.

I stress that the mere existence of supra-competitive prices in the simulations does not make the FEMs viable. In fact, the only reason supra-competitive prices arise in settings with function approximation is that *both* agents fail to compete efficiently. Indeed, their success hinges on the other agent also playing an inferior strategy. It is easy to see that playing such exploitable strategies are unlikely to succeed in real market settings. A potential exception is the *hub and spoke* scenario envisioned by Ezrachi and Stucke (2017). The exploitable algorithms could prove successful if a vendor was able to supply it to *all* competitors in an industry.

It is hard to pinpoint the exact causes of these failures. An obvious lever for improvement is the parametrization. The default specification was largely arbitrary and I did not systematically optimize parameters for computational reasons. The most obvious candidate for improving strategies is the exploration rate. But since the results are similar for a number of specifications, I am doubtful that optimizing parameters would make much of a difference. Alternatively, one could trial other, more sophisticated FEMs. However, linear function approximation might be generally inadequate to learn collusive strategies precisely because stable strategies require non-linear responses. I suspect that *linear* function approximation could be futile in the realm of multi-agent reinforcement learning in economic environments. Nevertheless, absence of evidence does not equate evidence of absence. Indeed, Hettich (2021) proves that agents learning with *non-linear* function approximation can be very successful in forging collusion.

I leave open two avenues for future research. First, further simulation studies could prove instrumental to understand under which conditions algorithmic collusion is likely. Most considered environments (including the one in this study) are rather simple and prefabricated. It would be interesting to see how algorithms behave in more challenging conditions (e.g. many players, dynamic demand, multi-sided markets). Possible extensions include *actor-critic models* that allow to incorporate continuous action spaces. Second, empirical studies on real markets are imperative to get a refined understanding of how real the threat of algorithmic collusion is. Assad et al. (2020) show that increased price margins in the wake of independently acquired algorithms are *possible*. Whether this results holds for other industries and over time remains to be seen.

References

Anderson, S. P. and A. de Palma (1992). "The Logit as a Model of Product Differentiation". *Oxford Economic Papers* 44(1), pp. 51–67. URL: https://www.jstor.org/stable/2663424.

Assad, S. et al. (2020). "Algorithmic Pricing and Competition: Empirical Evidence from the German Retail Gasoline Market". Working Paper. URL: https://papers.ssrn.com/abstract=3682021.

Boer, A. V. den (2015). "Dynamic Pricing and Learning: Historical Origins, Current Research, and New Directions". *Surveys in Operations Research and Management Science* 20(1), pp. 1–18. URL: http://www.sciencedirect.com/science/article/pii/S1876735415000021.

Bundeskartellamt (2021). *Working Paper - Algorithms and Competition*. URL: https://www.bundeskartellamt.de/SharedDocs/Publikation/EN/Berichte/Algorithms_and_Competition_Working-Paper.html.

Calvano, E. et al. (2020). "Artificial Intelligence, Algorithmic Pricing and Collusion". *American Economic Review* 110(10), pp. 3267–97. URL: https://papers.ssrn.com/abstract=3304991.

Chen, L., A. Mislove, and C. Wilson (2016). "An Empirical Analysis of Algorithmic Pricing on Amazon Marketplace". *Proceedings of the 25th International Conference on World Wide Web*, pp. 1339–1349. URL: https://doi.org/10.1145/2872427.2883089.

CMA (2016). *Case 50223*. URL: https://www.gov.uk/cma-cases/online-sales-of-discretionary-consumer-products.

Crandall, J. W. et al. (2018). "Cooperating with Machines". *Nature Communications* 9(1), p. 233. URL: https://www.nature.com/articles/s41467-017-02597-8.

DoJ (2017). "Algorithms and Collusion - Note by the United States". *OECD Roundtable on Algorithms and Collusion*.

EU (2017). "Algorithms and Collusion - Note from the European Union". *OECD Roundtable on Algorithms and Collusion*.

Ezrachi, A. and M. E. Stucke (2017). "Algorithmic Collusion: Problems and Counter Measures". *OECD Roundtable on Algorithms and Collusion*.

— (2018). "Sustainable and Unchallenged Algorithmic Tacit Collusion". *University of Tennessee Legal Studies Research Paper No. 366*. URL: https://papers.ssrn.com/abstract=3282235.

Gal, M. (2019). "Algorithms as Illegal Agreements". *Berkeley Technology Law Journal* 34(1), pp. 67–118. URL: https://papers.ssrn.com/abstract=3171977.

Gal, M. and N. Elkin-Koren (2017). "Algorithmic Consumers". *Harvard Journal of Law and Technology* 30. URL: https://papers.ssrn.com/abstract=2876201.

Harrington, J. E. (2018). "Developing Competition Law for Collusion by Autonomous Artificial Agents". *Journal of Competition Law & Economics* 14(3), pp. 331–363. URL: https://doi.org/10.1093/joclec/nhy016.

Hastie, T., R. Tibshirani, and J. Friedman (2009). "Basis expansions and regularization". *The Elements of Statistical Learning*. Springer, pp. 139–189.

Hettich, M. (2021). "Algorithmic Collusion: Insights from Deep Learning". URL: https://papers.ssrn.com/abstract=3785966.

Ittoo, A. and N. Petit (2017). "Algorithmic Pricing Agents and Tacit Collusion: A Technological Perspective". *L'intelligence artificielle et le droit*. Social Science Research Network, pp. 241–256. URL: https://papers.ssrn.com/abstract=3046405.

Jaakkola, T., M. I. Jordan, and S. P. Singh (1994). "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms". *Neural Computation* 6(6), pp. 1185–1201. URL: https://doi.org/10.1162/neco.1994.6.6.1185.

Johnson, J., A. Rhodes, and M. R. Wildenbeest (2020). "Platform Design When Sellers Use Pricing Algorithms". URL: https://papers.ssrn.com/abstract=3691621.

Kimbrough, S. and F. Murphy (2009). "Learning to Collude Tacitly on Production Levels by Oligopolistic Agents". *Computational Economics* 33(1), pp. 47–78.

Klein, T. (2021). "Autonomous Algorithmic Collusion: Q-Learning Under Sequential Pricing". *The RAND Journal of Economics* forthcoming. URL: https://onlinelibrary.wiley.com/doi/10.1111/1756-2171.12383.

Leibo, J. Z. et al. (2017). "Multi-agent Reinforcement Learning in Sequential Social Dilemmas". *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems*, pp. 464–473.

Maskin, E. and J. Tirole (1988). "A Theory of Dynamic Oligopoly, II: Price Competition, Kinked Demand Curves, and Edgeworth Cycles". *Econometrica* 56(3), pp. 571–599. URL: https://www.jstor.org/stable/1911701.

Mehra, S. K. (2015). "Antitrust and the Robo-Seller: Competition in the Time of Algorithms". *Minnesota Law Review* 100, p. 1323. URL: https://papers.ssrn.com/abstract=2576341.

Mnih, V. et al. (2015). "Human-level control through deep reinforcement learning". *Nature* 518, pp. 529–533. URL: https://www.nature.com/articles/nature14236.

Motta, M. (2004). *Competition Policy: Theory and Practice*. Cambridge University Press.

Naik, A. et al. (2019). "Discounted Reinforcement Learning Is Not an Optimization Problem". *Preprint*. URL: http://arxiv.org/abs/1910.02140.

Noel, Michael D. (2008). "Edgeworth Price Cycles and Focal Prices: Computational Dynamic Markov Equilibria". *Journal of Economics & Management Strategy* 17(2), pp. 345–377. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1530-9134.2008.00181.x.

OECD (2016). "Price discrimination - Background Note by the Secretariat". URL: https://www.oecd.org/daf/competition/price-discrimination.htm.

— (2017). "Algorithms and Collusion: Competition Policy in the Digital Age". *Roundtable on Collusion and Algorithms*. URL: http://www.oecd.org/competition/algorithms-collusion-competition-policy-in-the-digital-age.htm.

oefgen (2019). *Decision of 26.07.2019*. URL: https://www.ofgem.gov.uk/publications-and-updates/investigation-whether-economy-energy-e-gas-and-electricity-and-dyball-associates-have-infringed-chapter-i-competition-act-1998-respect-suspected-anti-competitive-agreement.

Precup, D., R. S. Sutton, and S. P. Singh (2000). "Eligibility Traces for Off-Policy Policy Evaluation". *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 759–766.

Rummery, G. and M. Niranjan (1994). "On-Line Q-Learning Using Connectionist Systems". URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.2539.

Schwalbe, Ulrich (2018). "Algorithms, Machine Learning, and Collusion". *Journal of Competition Law & Economics* 14(4), pp. 568–607. URL: https://papers.ssrn.com/abstract=3232631.

Schwartz, Anton (1993). "A Reinforcement Learning Method for Maximizing Undiscounted Rewards". *Proceedings of the 10th International Conference on Machine Learning*, pp. 298–305.

Seijen, H. and R. Sutton (2014). "True Online TD(lambda)". *International Conference on Machine Learning*, pp. 692–700. URL: http://proceedings.mlr.press/v32/seijen14.html.

Sutton, R. S. (1988). "Learning to Predict by the Methods of Temporal Differences". *Machine Learning* 3(1), pp. 9–44. URL: https://doi.org/10.1007/BF00115009.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning, second edition: An Introduction*. MIT Press.

Tsitsiklis, J. N. and B. Van Roy (1997). "An analysis of Temporal-Difference Learning with Function Approximation". *IEEE Transactions on Automatic Control* 42(5), pp. 674–690.

Tuyls, Karl and Gerhard Weiss (2012). "Multiagent Learning: Basics, Challenges, and Prospects". *AI Magazine* 33(3), p. 41. URL: https://www.academia.edu/15087104/Multiagent_learning_Basics_challenges_and_prospects.

Waltman, L. and U. Kaymak (2008). "Q-learning agents in a Cournot oligopoly model". *Journal of Economic Dynamics and Control* 32(10), pp. 3275–3293. URL: https://econpapers.repec.org/article/eeedyncon/v_3a32_3ay_3a2008_3ai_3a10_3ap_3a3275-3293.htm.

Appendix A Function Approximation Methods

This section describes the function approximation methods and their parametrization (see Table 2) in more detail. Note that *polynomial approximation*, as described in section A.2, is not directly used in this study but nevertheless introduced as a precursor to the final two methods *polynomial tiles* and *separate polynomials*.

A.1 Tile coding

In reinforcement learning, *tile coding* is a common way to extract binary features from a state-action space.[41] Its appeal stems partly from the fact that it is a generalization of tabular learning. The idea is that several *tilings* superimpose the state-action space. The $\mathcal{T}$ tilings are offset but each tiling covers the entire state-action space:

$$T^L \leq A^L \text{ and } T^U \geq A^U \quad \forall \quad T \in \{1, 2, ..., \mathcal{T}\} \quad , \tag{23}$$

where T^L and T^U, respectively, represent the lower and upper bound of tiling T. Each tiling is itself composed of uniformly spaced out *tiles*.[42] Every tile is uniquely demarcated by a lower and an upper threshold for every dimension. Consequently, the number of tiles per tiling is controlled by the number of thresholds. For this simulation, it suffices to define a single set of thresholds per tiling that applies to all 3 dimensions. More specifically, the thresholds are spaced out evenly in the tiling-specific interval $[T^L, T^U]$:

$$\{T^L, T^L + \frac{1(T^U - T^L)}{\psi - 1}, T^L + \frac{2(T^U - T^L)}{\psi - 1}, ..., T^L + \frac{(\psi - 2)(T^U - T^L)}{\psi - 1}, T^U\} \quad , \tag{24}$$

where ψ represents the number of thresholds. This gives rise to $(\psi - 1)^3$ tiles per tiling. As indicated, tiles are binary, i.e. if a state-action observation falls into a particular demarcation, the corresponding tile is *activated*:

$$x_d^{Tile\ Coding} = \begin{cases} 1 & \text{if } \{p_{i,t-1}, p_{j,t-1}, p_{i,t}\} \text{ in tile demarcation}_d \\ 0 & \text{if } \{p_{i,t-1}, p_{j,t-1}, p_{i,t}\} \text{ not in tile demarcation}_d \end{cases} \tag{25}$$

Since tiles within a tiling are non-overlapping, any state-action combination activates exactly $\mathcal{T}$ tiles, one per tiling. The total number of features is simply $T\ (\psi - 1)^3$. Note that the tabular case can be recovered by setting $\mathcal{T} = 1$ and $\psi \geq m + 1$. In this case, every tile is activated by at most one feasible state-action combination which is equivalent

[41] For an extensive introduction with helpful illustrations refer to Sutton and Barto (2018, pp.217-221).

[42] With 2 dimensions, a tiling simply corresponds to a 2-dimensional grid. In our case, the state-action space is 3-dimensional, so it may prove more intuitive to think of cubes instead of tilings and tiles.

to storing a dedicated coefficient for every state-action combination.[43] For this study, I set $\mathcal{T} = 5$ and $\psi = 9$. These parameters give rise to $2,560$ elements in $\boldsymbol{w}$, about a third of the size in tabular learning.

A.2 Polynomials

Polynomial approximation applies polynomial transformations to its inputs. In order to keep this (and the upcoming) section brief, I will introduce the notation for the specific case of 3 variables.[44] Polynomial approximation of order k maps S_t and A_t to a set of features, where a single feature corresponds to:

$$x_d^{Poly} = p_{i,t-1}^{\kappa_{d,1}} \; p_{j,t-1}^{\kappa_{d,2}} \; p_{i,t}^{\kappa_{d,3}} \tag{26}$$

Every combination of exponents that adheres to the restrictions

- $0 < \kappa_{d,1} + \kappa_{d,2} + \kappa_{d,3} \leq k \quad \forall \; d$ and
- $\kappa_{d,1}, \kappa_{d,2}, \kappa_{d,3} \in \{0, 1, ..., k\} \quad \forall \; d$

constitutes one feature. Using polynomial approximation, the feature vector $\boldsymbol{x}$ contains $\binom{k+3}{3} - 1$ elements. I choose not to use a simple polynomial to approximate the valuation of the entire state-action space. Exploratory runs have shown that the method has trouble converging and frequently produces unreasonable results in the provided environment. Perhaps, this is not surprising because every state-action combination will always produce non-zero values for *all* features and change every single element in $\boldsymbol{w}$. While this facilitates learning similarity of actions, it makes it more difficult for the algorithm to develop distinct notions for very different prices.

A.3 Polynomial tiles

What I call *polynomial tiles* is a blend of *tile coding* and *polynomial approximation*. To be precise, just as in tile coding, the state-action space is divided into overlapping tiles. However, instead of a binary indication, every tile comprises a distinct polynomial. For the sake of notation, it is helpful to divide the index d into a tiling component e and a

[43] If $\psi > m + 1$, some tiles would never be activated. Conversely, every state-action combination would correspond to a unique tile.

[44] For a more thorough treatment with variations and extensions, see e.g. Hastie, Tibshirani, and Friedman (2009).

polynomial part f. Hence:

$$x_d^{Poly\ Tiles} = x_{e,f}^{Poly\ Tiles} = \begin{cases} p_{i,t-1}^{\kappa_{f,1}}\ p_{j,t-1}^{\kappa_{f,2}}\ p_{i,t}^{\kappa_{f,3}} & \text{if } \{p_{i,t-1}, p_{j,t-1}, p_{i,t}\} \text{ in tile demaraction}_e \\ 0 & \text{if } \{p_{i,t-1}, p_{j,t-1}, p_{i,t}\} \text{ not in tile demarcation}_e \end{cases} \tag{27}$$

The restrictions on the exponents κ from Appendix A.2 apply. The method accompanies $T\ (\psi-1)^3(\binom{k+3}{3}-1)$ features. As this method allows for a distinguished value estimation for different state-action combinations even within a tile, it appears reasonable to increase the size of the tiles in order to decrease the number of coefficients and avoid overfitting. Specifically, I retain the number of tilings ($\mathcal{T} = 5$), but reduce the number of tiles per tiling from 512 to 64 by imposing $\psi = 5$. Moreover, I allow for polynomial combinations up to degree $k = 4$. With a total number of $10,880$ parameters, $\boldsymbol{w}$ is about 50% larger than in tabular learning.

A.4 Separate polynomials

Separate polynomials maintain for every action a distinct set of parameters that apply *polynomial approximation* to the state set. In reinforcement learning, it is common to store a separate set of coefficients for every feasible action.[45] Since A_t is fixed within each set, the polynomial only considers S_t:[46]

$$x_d^{Separate\ Poly} = \begin{cases} p_{i,t-1}^{\kappa_{d,1}}\ p_{j,t-1}^{\kappa_{d,2}} & \text{if } a = A_t \\ 0 & \text{if } a \neq A_t \end{cases} \tag{28}$$

The number of encompassed features arises naturally as $m(\ \binom{k+2}{2}\ -1)$.

Note that the method models the value of an action as a function of S_t. Perhaps, an inverse variation could be more intuitive from an economic perspective. Consider that every permutation of S_t holds a distinct set of parameters. This approach is closer to the notion of optimizing *a given* a fixed state set s. I leave this variation open as a potential avenue for future research.

[45] This approach is best suited if the action space is qualitative and the state space continuous. In this simulation, only the latter is strictly true. Therefore, the two issues inherent to tabular learning I have outlined in section 2.3.1, also apply to the action space of *separate polynomials*, but not to the state space.

[46] The restrictions on k are adjusted accordingly:

- $0 < \kappa_{d,1} + \kappa_{d,2} \leq k \quad \forall d$ and
- $\kappa_{d,1}, \kappa_{d,2} \in \{0, 1, ..., k\} \quad \forall d$

Appendix B Further Results

B.1 Cycle length and price range

Figure 10 offers some interesting insights on the distribution of cycle length. Unsurprisingly, a first glance suggests that the frequency of runs decreases with cycle length. Not accounting for differences between selection methods, the bars appear similar to a geometric distribution with the largest bar corresponding to a 'cycle length of 1' (i.e. no cycle at all). Moving towards the right, the frequency of observed runs decreases with cycle length, though at a decreasing pace. In fact, there are 6 runs with the largest considered cycle length of 10.

The differences between FEMs are substantial. Polynomial tiles largely follows the described decaying pattern. Similarly, tile coding rarely converges in long cycles, though the most prevalent cycle length is 2 (with a total of 220 runs). Interestingly, the frequency of cycle length of converged tabular runs is distributed almost uniformly. This observation also suggests that the employed convergence rule may well have misclassified some of the runs in the top left panel of Figure 1 as *not converged* where in reality the convergence cycle length simply exceeded the threshold arbitrarily set at 10. Finally, all runs of the separate polynomials FEM converged in a static price. This is in sharp contrast to the other methods and reinforces the conjectures from section 3.3. It is not obvious why there exist such differences between FEMs, especially since there is no economic justification for price cycles in a simultaneous pricing environment in the first place.

The observation that function approximation FEMs tend to converge with lower cycle lengths is extended with the insight that those methods also produce lower prices ranges. Figure 11 plots the range between the lowest and highest price a single agent charges in a cycle upon convergence. Naturally, the price range is null if an agent does not vary its actions at all. Perhaps unsurprisingly, the price range then tends to increase with cycle length, at times becoming remarkably high. The range of prices due to tabular learning frequently exceeds the range between collusive and Nash prices. This is a clear indication that price setting is occasionally irrational. Irrespective of agents competing or colluding, prices outside this range are not economically optimal. Note however, the inversion of the argument is dangerous. One should not deduct that behavior is *closer to optimal* from the mere fact that prices appear more stable. In fact, the study shows that the strategies learned with function approximation are often far from optimal and easy to exploit.

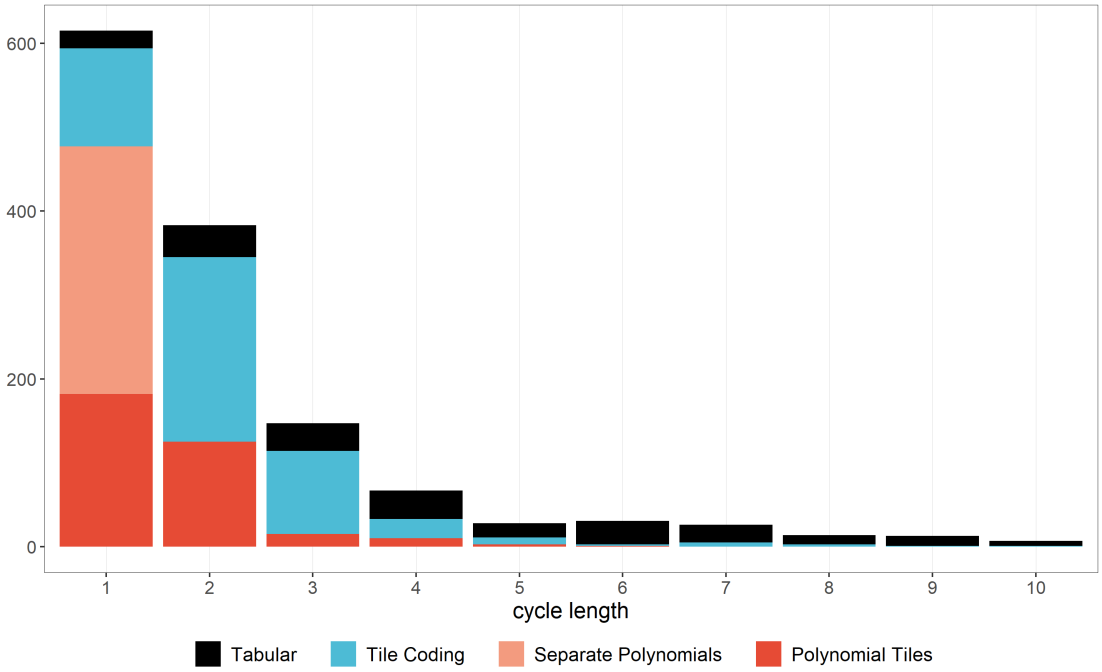

Figure 10: Count of converged runs by FEM and cycle length.

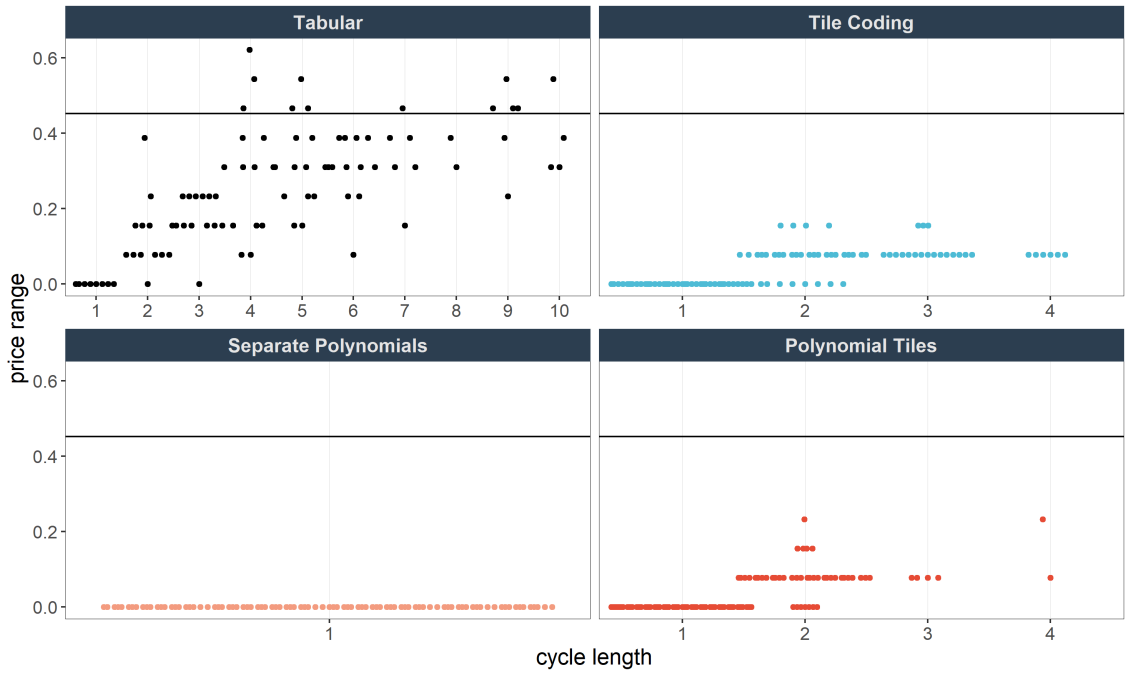

Figure 11: Distribution of price range by FEM and cycle length. Every point represents a single run. Within groups, points are spaced out horizontally. Price range is defined as the difference between the highest and lowest price an agent charges within a cycle. Only converged runs are considered (as cycle length is unavailable for other runs). Horizontal line represents the difference between collusive and Nash outcome (i.e. $p_m - p_n$).

B.2 Distribution of Δ

Section 3.2 illustrates that mean profits across experiments exceed Nash prices. To show that this is not an artifact of averaging, Figure 12 displays a violin plot illustrating the distribution of Δ per experiment. The distribution largely confirms the conclusion that most runs converge between Δ_m and Δ_n. The only FEM that generated a significant quantity of runs with profits below the Nash benchmark are separate polynomials where 39.3% of runs converged with profits below the Nash equilibrium (though this is heavily biased through the experiments with high α where the share of negative average Δ is 100%). While the other FEMs tend to elicit runs within the benchmarks, the variability remains quite high. This indicates a degree of path dependence and suggests that the algorithms are prone to stick to early explored strategies that are *above average*, but *sub-optimal*. Polynomial tiles exhibit the narrowest range of Δ, in particular for low α.

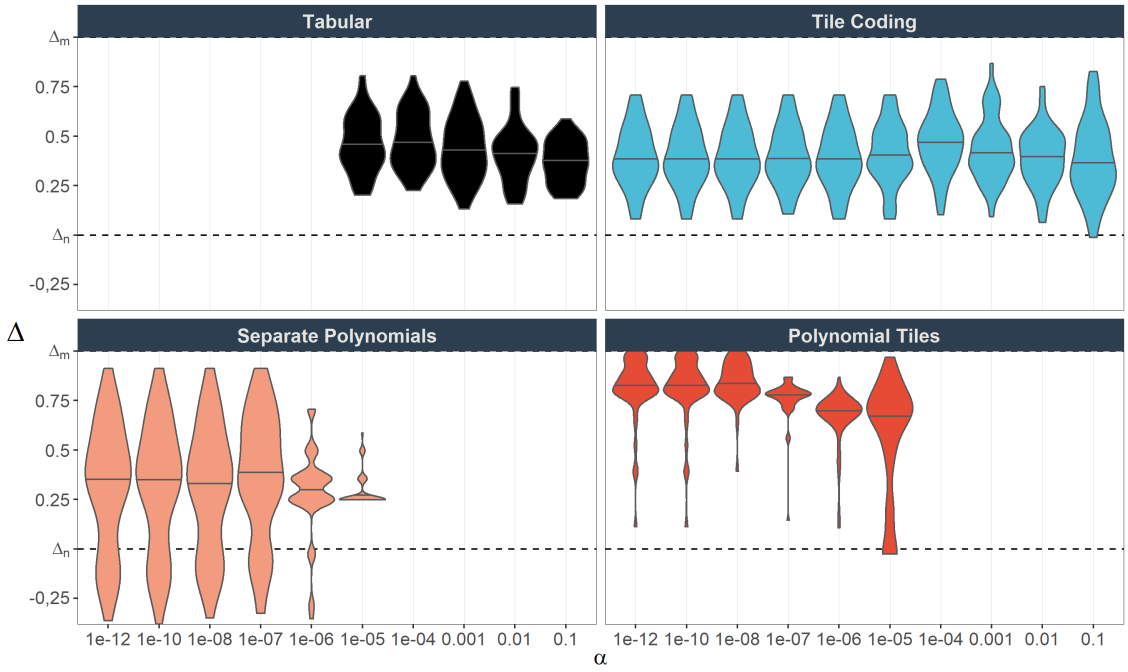

Figure 12: Distribution of Δ by FEM and α. Includes converged and non-converged runs. Violin widths are scaled to maximize width of individual violins, comparisons of widths between violins are not meaningful. Violins are trimmed at smallest and largest observation respectively. Three experiment (polynomial tiles with $\alpha = 0.0001$ and separated polynomials with $\alpha \in \{0.001, 0.0001\}$) and a single run with $\Delta < -0.5$ are excluded for better presentability. Horizontal lines represent the median.

B.3 Price trajectory

Figure 13 displays the trajectory of Δ over time. Only runs of the *optimized* experiments are printed, as explained in Table 4. Δ is averaged over 50,000 periods apiece and over both players. Due to amassed exploration early in the simulation, average profits are low early on but increase over time. Furthermore, starting at $t = 250,000$ violin widths decrease because of some runs triggering the convergence criteria. Interestingly, the non-converging runs in the optimized separate polynomials experiments are characterized by profits *below* the static Nash equilibrium. Note also the remarkable speed at which polynomial tiles increases profits. After merely $100,000$ periods, the median Δ hovers around 0.75 already.

Figure 14 depicts the distribution of average *prices* in optimized experiments over time and Figure 15 displays the price and profit trajectory of single runs over time. Both figures illustrate that, by and large, prices and profits remain within the benchmarks of Nash competition and the fully collusive case. Obviously, this does not apply to every single period, but holds true on average.

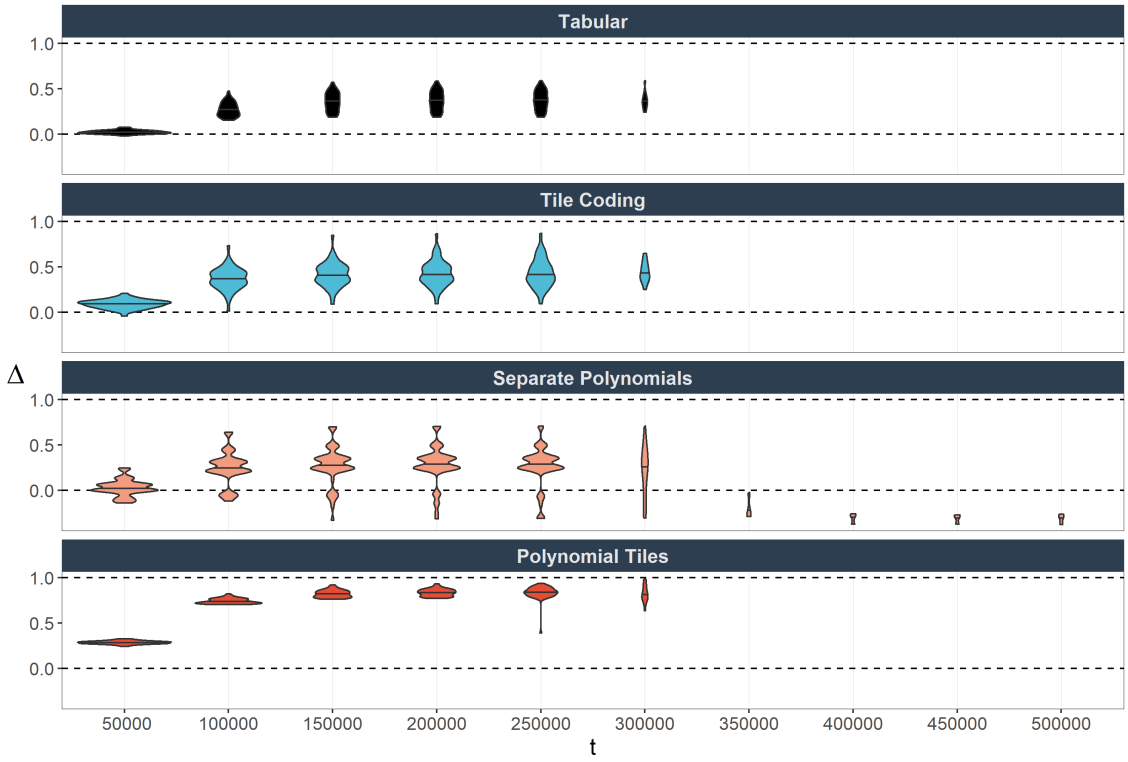

Figure 13: Distribution of trajectory of Δ by FEM with optimized α (see Table 4). For individual runs, Δ is averaged over 50,000 periods apiece and both players. Plot includes converged and non-converged runs. Violin widths represent quantity of active runs at t which enables comparisons between violins. As most runs converge after 200,000 to 300,000 periods, violin widths decrease thereafter. Violins are trimmed at smallest and largest observation respectively. Horizontal lines represent the median.

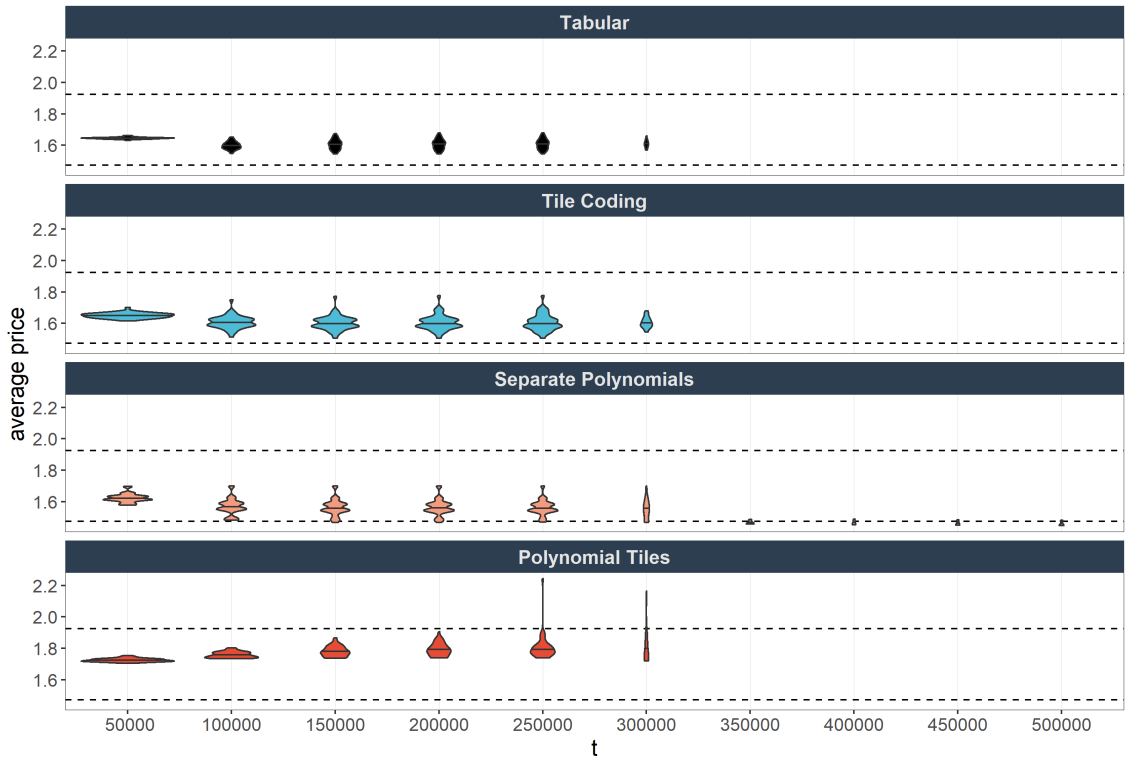

Figure 14: Distribution of trajectory average prices by FEM with optimized α. For individual runs, p is averaged over 50,000 periods apiece and both players. Plot includes converged and non-converged runs. Violin widths represent quantity of active runs at t which enables comparisons between violins. As most runs converge after 200,000 to 300,000 periods, violin widths decrease thereafter. Violins are trimmed at smallest and largest observation respectively. Horizontal lines represent the median.

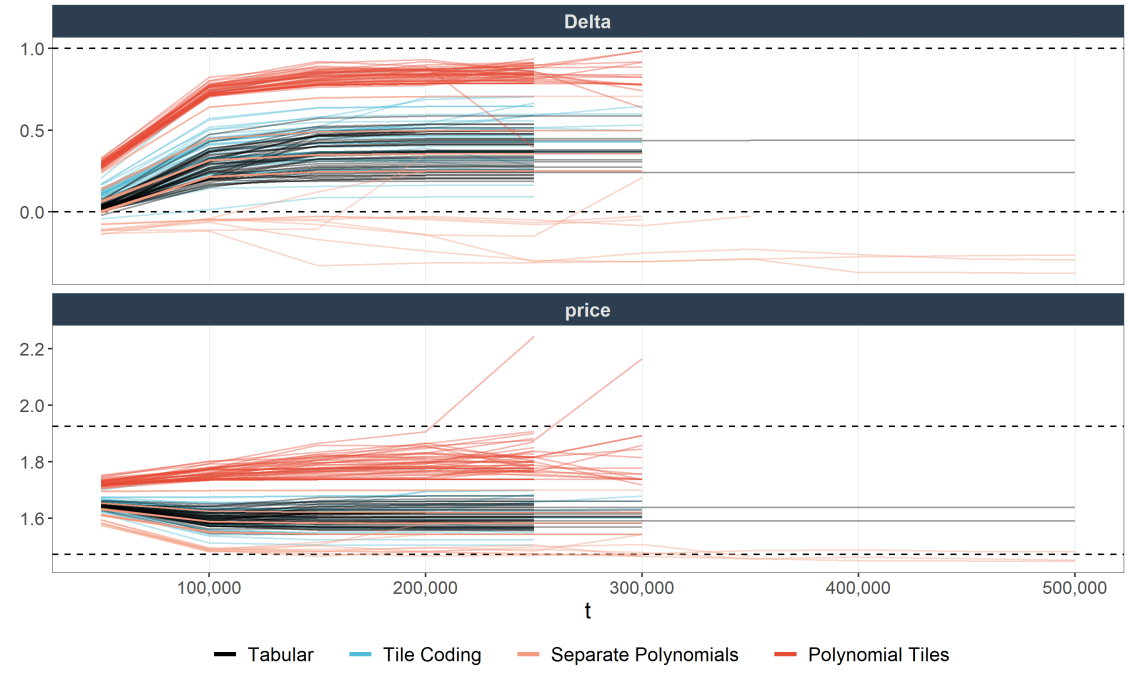

Figure 15: Trajectories of average Δ (top panel) and prices (bottom panel) with optimized α. For individual runs, the respective metric is averaged over 50,000 periods apiece and both players. Plot includes converged and non-converged runs.

B.4 Deviations

Figure 16 displays the difference between profits agents receive after the forced deviation takes place and profits of the alternative path with no deviation. Naturally, the deviating agent makes larger profits at the deviation period $\tau = 1$. With tabular learning, her profits decrease thereafter due to retaliatory prices. The plot shows that this occasionally happens with tile coding and polynomial tiles. With all FEMs, profits tend to return to pre-deviation levels quickly.

Figure 17 displays a frequency polygon to gauge how much more or less profitable the deviation is compared to the counterfactual of sticking to the learned strategy. With tabular learning, most deviations end up being unprofitable. Contrary, the line for polynomial tiles indicates that most deviations are profitable, some of them quite high. With regards to tile coding and separated polynomials, the deviations in many runs seem to yield profits similar to not deviating. Unsurprisingly, the bottom panel is skewed to the left suggesting that the deviation experiment is unprofitable for the non-deviating agent.

Figure 18 depicts the price trajectory of three individual runs belonging to the optimized polynomial tiles experiment. Run *06* shows the deviating agent continuing to cheat and increasing her long-term profits. It also shows that price asymmetries *between* players are not an uncommon phenomenon. Run *08* shows the non deviating agent meeting the price cut. This culminates in a new equilibrium with lower prices. Since the pre-deviation prices were slightly above the collusive benchmark, the new equilibrium actually improves both agent's profits in this particular example. Run *09* shows the common pattern of quick reversal to pre-deviation prices and profits.

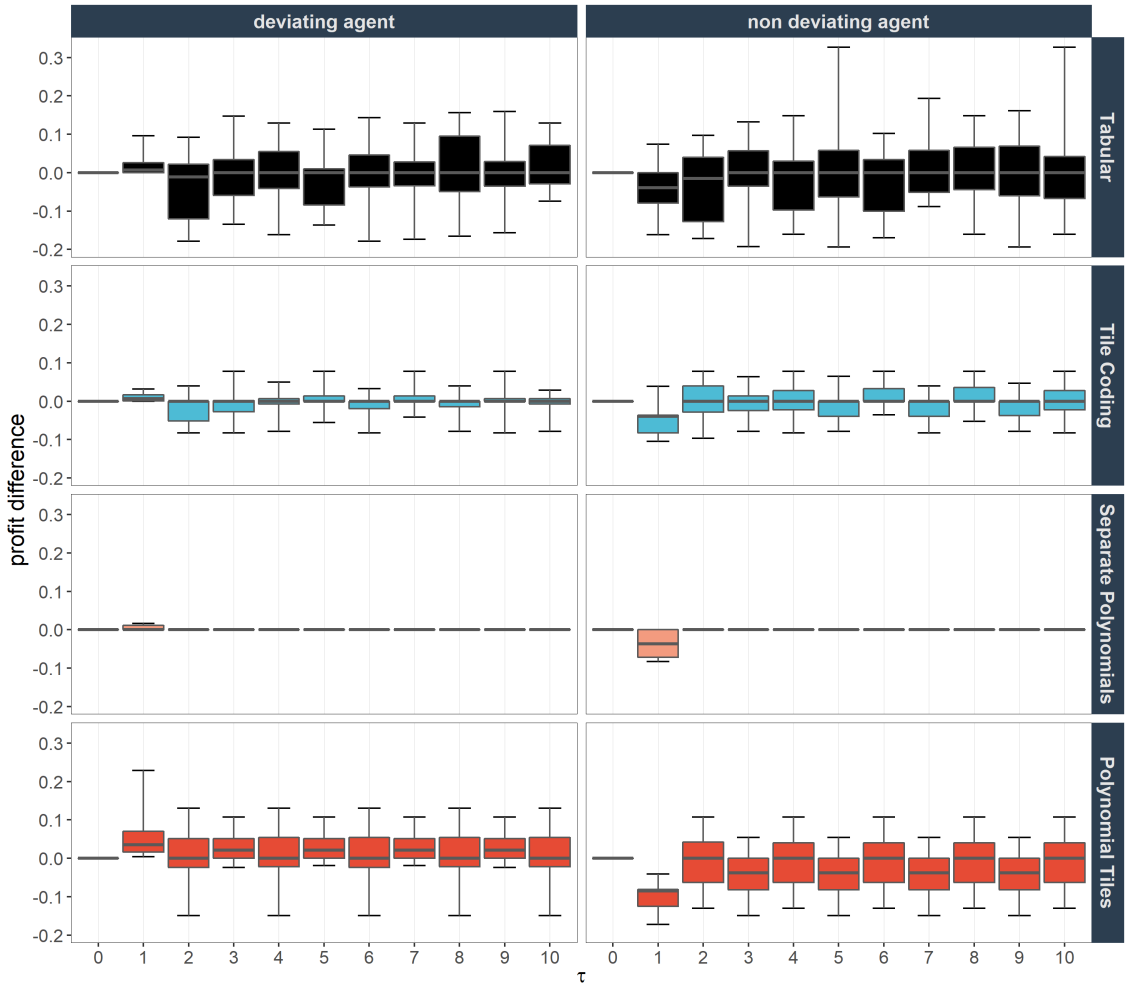

Figure 16: Distribution of profit differences around deviation relative to counterfactual path *without* forced deviation, i.e. the difference to the price had no deviation taken place, by FEM. Only includes converged runs because a clear counterfactual exists. Boxes demarcate 15th and 85th percentiles. They are extended by whiskers that mark the entire range of price differences. Horizontal lines represent the group median.

Figure 17: Distribution of additional profitability due to deviation by agent and FEM. Width of bins: 0.02. 13 extreme data points (< -0.5 or > 0.5) are excluded for better presentability. Deviations are deemed *profitable* if the discounted profits until $\tau = 10$ due to the deviation exceed profits from a counterfactual without deviation. Only includes converged runs because a clear counterfactual exists. Discounting is equivalent to γ in (13), i.e. 0.95. A significant number of deviations are neither profitable nor unprofitable. In those runs, the learned strategy of the deviating agent is actually the best response at $\tau = 1$ and both agents keep following their respective price cycle.

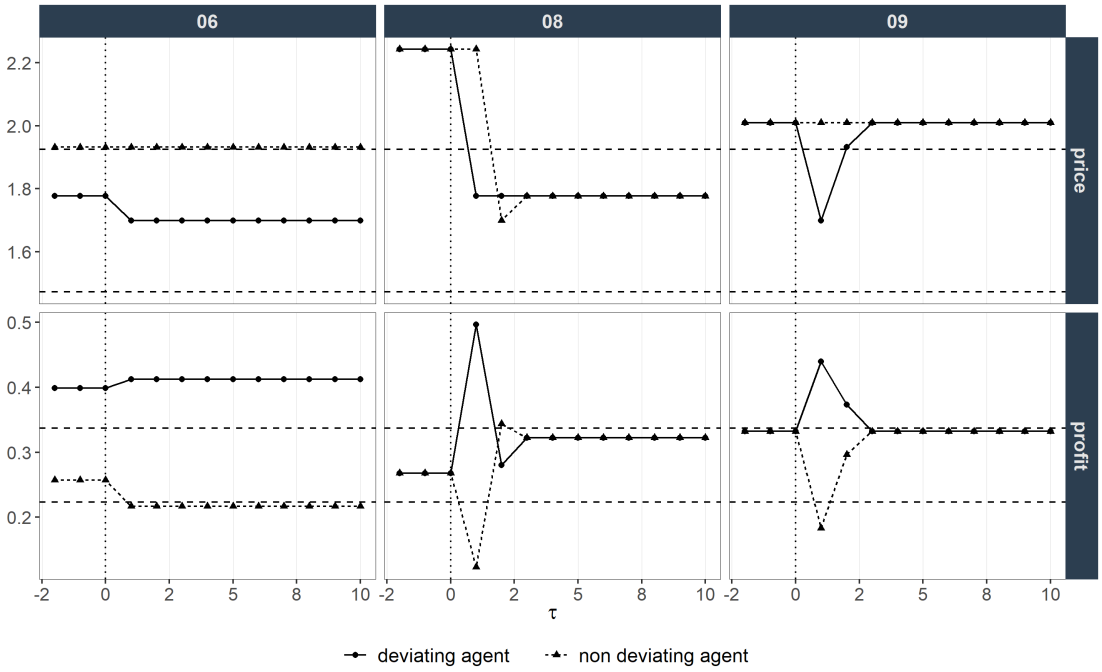

Figure 18: Prices and profits in 3 exemplary polynomial tiles deviation experiments with optimized α. Top panels display prices, bottom panels profits. The exemplary runs are stacked horizontally . Numbers on strip indicate assigned run id. Dashed horizontal lines represent the fully collusive and static Nash benchmarks. Dotted vertical line reflects time of convergence, i.e. the period immediately before the forced deviation.

B.5 Prolonged deviations

To extend the mixed results from the deviation experiments in section 3.3, I conducted another *prolonged deviation* experiment with continued learning. As I will show, the previously drawn conclusions remain intact. However, first I briefly explain why a continued intervention could theoretically be different from a one-time deviation. The critical components are continued learning and the eligibility trace vector $\boldsymbol{z}$ that make conceivable an agent tolerating isolated deviations but punishing longer price cuts.

Without eligibility traces and the ability to learn, a one-time deviation suffices to assess retaliatory behavior because the memory is too short to *remember* that the opponent cheated for longer than a single period. Likewise, stripping the non deviating agent of his ability to update $\boldsymbol{w}$ renders him unable to learn that tolerating deviations is exploitable and can culminate in continuous low rewards. Consequently, if he failed to punish a deviation at $\tau = 2$, he will not react at $\tau = 3$ either. On the contrary, with the ability to learn enabled, both agents can readjust the parameter updates. For instance, after discovering that tolerating a one-time deviation yields a low reward, the non deviating agent might adjust $\boldsymbol{w}$ and decide to play a different action next time he is cheated (e.g. match the price cut). This is augmented by the length of deviation episodes and the existence of eligibility traces. If the deviating agent *continues* to cheat, the opponent should continue to decrease the valuation of the *tolerating* strategy and could ultimately fall back to the next best action (which might be a price cut).[47]

The *prolonged deviation* experiment lasts 20 periods in total. It was set up as follows. The deviating agent anticipates the price of her opponent perfectly and continuously plays the best response for a total of 10 periods of cheating. This imposes the assumption that she is capable of perfectly predicting her opponent's response to the initial deviation. Exploration remains disabled ($\epsilon = 0$) but both agents continue learning from their actions.[48] After I stop forcing the deviating agent to play the best response, both agents play another 10 periods adhering to their learned strategies.

The additional deviation experiments were conducted with the optimal values of α in accordance with Table 4. Figure 19 displays the average price trajectory around the prolonged deviation. Figure 20 provides a more detailed view on the distribution of responses. Both plots confirm the previous observations. Only with tabular learning does

[47] Furthermore, remember that the deviation experiment is conducted right after convergence was detected. Consequently, the algorithm was *on-path* for a large number of periods and the eligibility traces have not been reset recently (see (14)). Therefore, updates are large in magnitude and after the deviation experiment concludes, the convergence equilibrium might not be feasible anymore because its valuation by the agents changed.

[48] I also prescribe that the forced deviation is considered *on-policy*. Since $\epsilon = 0$, this is most natural to incorporate.

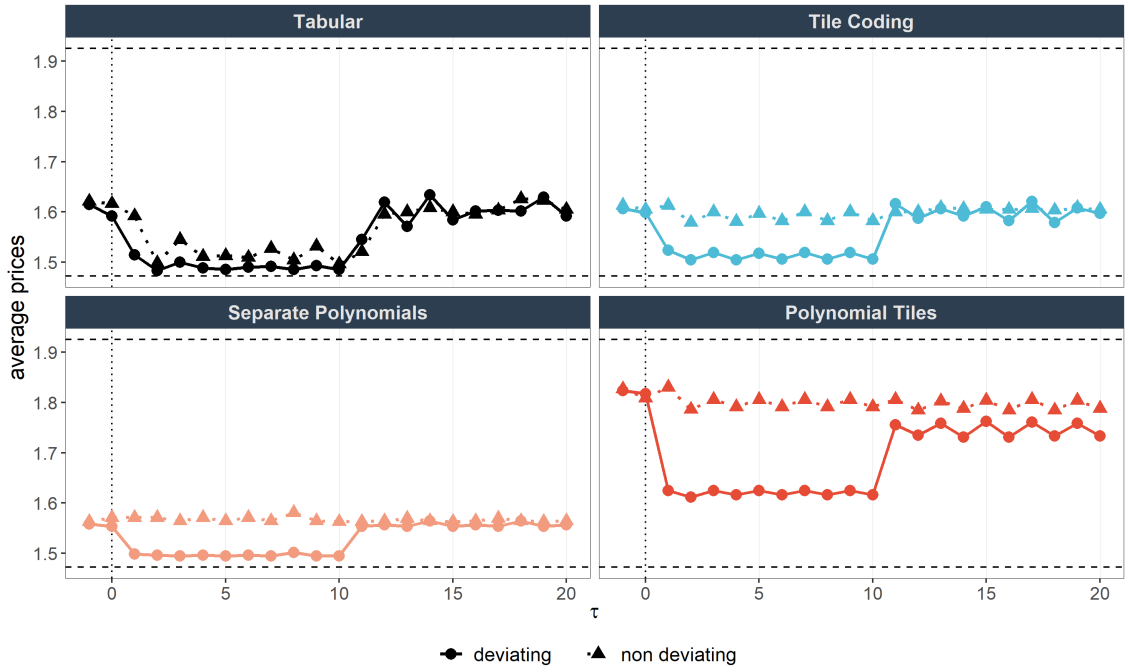

Figure 19: Average price trajectory around prolonged deviation by FEM. Points represent the average price over all runs of an experiment. Dashed horizontal lines represent the fully collusive price p_m and the static Nash solution p_n. Dotted vertical line reflects time of convergence, i.e. the period immediately before the forced deviation.

the non deviating agent match the price cuts systematically. The top left panel shows both agents hover around the Nash benchmark for the entire duration of the deviation episode. Clearly, this is unprofitable for both agents but the punishment is necessary to sustain supra-competitive prices in the first place. Note also the quick return to pre-deviation levels as soon as the deviating agent returns to her learned behavior. Since the opponent follows immediately, it appears that the return must be initiated be the original cheater. The experiment illustrates that the supra-competitive outcomes remain sustainable in the face of persistent interruptions.

With regard to the three function approximation methods, the deviating agent appears to systematically exploit her opponent who fails to punish the price cut. The subtle differences between FEMs extend to the prolonged deviation. Separate polynomials evoke no response from the non deviating agent. Both tiling methods show a small *average* price cut over the duration of the prolonged deviation, but this response falls short of a reliable mechanism that consistently deters deviation across runs. Indeed, Figure 20 shows that only isolated runs exhibit the non deviating agent cutting the pre-deviation price levels. Most runs show no reaction which is veiled by averaging over all runs of the experiment. This absence of a retaliation opens up the opportunity for continuous exploitation by the deviating agent. Despite that, the latter tends to return to pre-

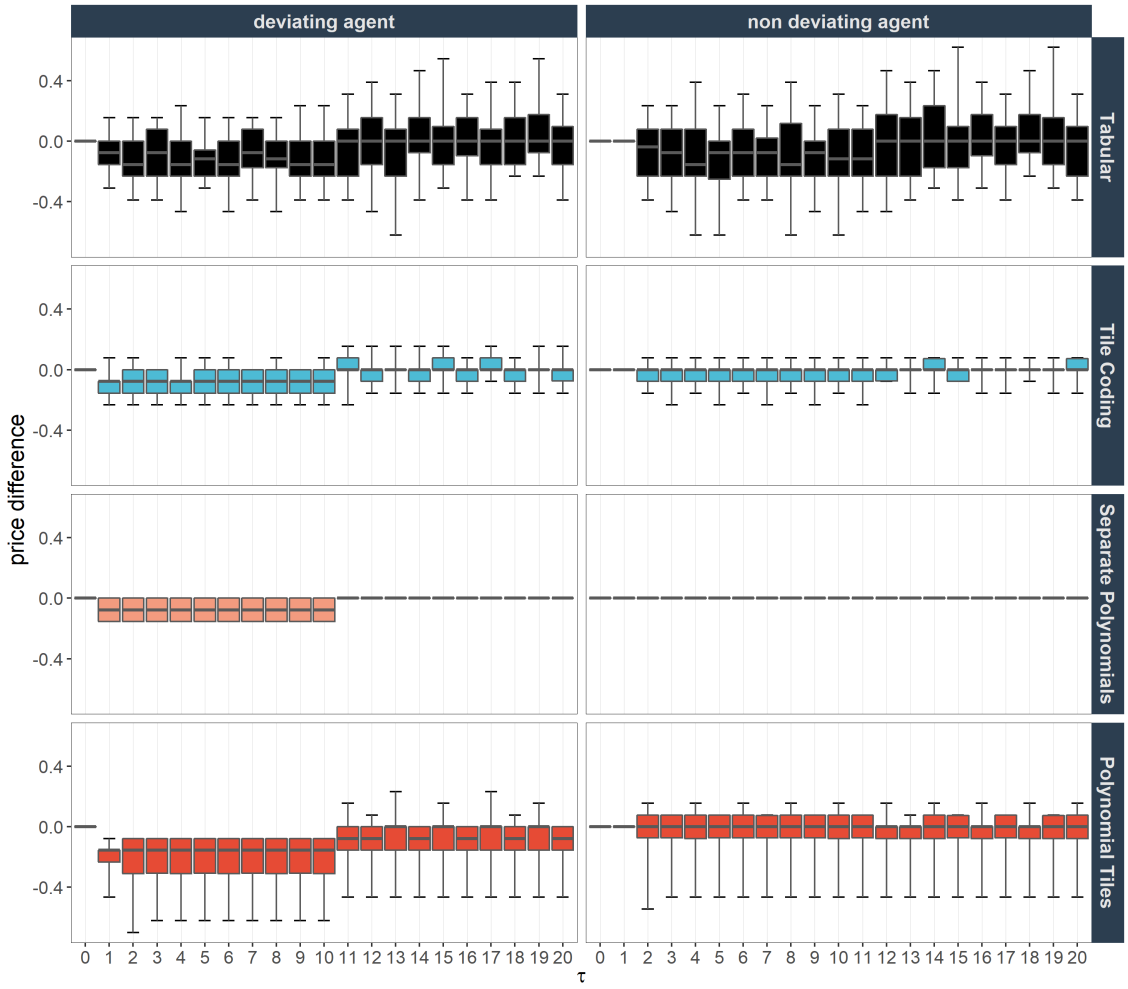

Figure 20: Distribution of price differences around prolonged deviation relative to counterfactual path *without* forced deviation, i.e. the difference to the price had no deviation taken place, by FEM. Only includes converged runs because a clear counterfactual exists. Boxes demarcate 15th and 85th percentiles. They are extended by whiskers that mark the entire range of price differences. Horizontal lines represent the group median.

deviation price levels. Therefore, both agents act far from optimal (in the economic sense of the word) and fail to learn (enough) from the prolonged deviation experiment. Lastly, note the difference between pre- and post-deviation price levels at the bottom right panel, representing polynomial tiles. As noted previously, this suggests that the agents proceed to play a different, less profitable equilibrium after the deviation. This easy switch to a new strategy further challenges the viability of the pre-deviation equilibrium in the first place.

It is conceivable, maybe even likely, that the non deviating agent does alter its strategy after a time frame much longer than 10 periods. However, this is not important for this study because the agent's strategy is easy to exploit in the short term and the deviations are clearly profitable (refer to Table 5).

Appendix C Further Variations

C.1 Exploration (β)

Section 4.1 showed that deviations in tabular learning environments were less likely to be profitable if exploration was extensive. Figure 21 confirms that the convergence equilibria are more likely to be underpinned by severe punishment strategies if β decreases (i.e. exploration becomes more extensive). The immediate response of the non deviating agent is harshest with $\beta = 10^{-5}$.

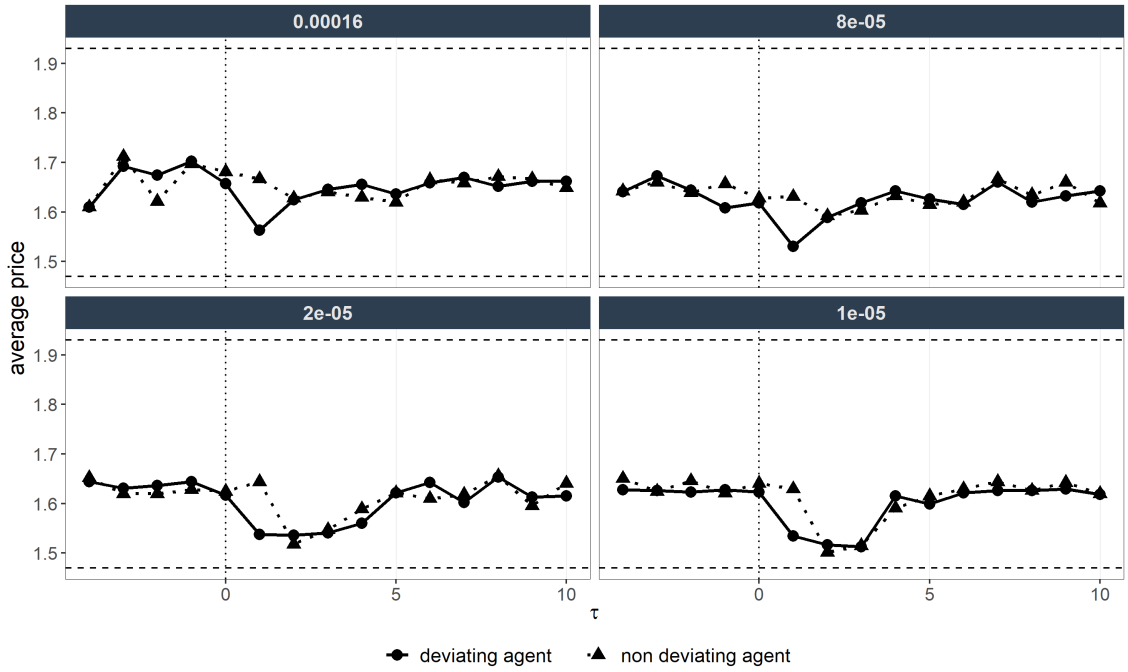

Figure 21: Average price trajectory around deviation by β (values on strip). Includes only tabular learning experiments. Points represent the average price over all runs of an experiment. Dashed horizontal lines represent the fully collusive price p_m and the static Nash solution p_n. Dotted vertical line reflects time of convergence, i.e. the period immediately before the forced deviation.

C.2 Memory (λ)

Recall that high values of λ increase the algorithm's hindsight but also the variance. This is reflected in both convergence rates and outcomes. Figure 22 clearly indicates that high values of λ impede convergence for tabular learning and the separate polynomials FEM. Similarly, Figure 23 exhibits greater variability in profits with increasing λ. This holds true for all feature extraction methods, but is most salient for separate polynomials where a significant number of runs end in profits below the Nash equilibrium once λ exceeds 0.6.[49]

[49] The runs with $\Delta < 0$ are mainly runs where convergence was not achieved.

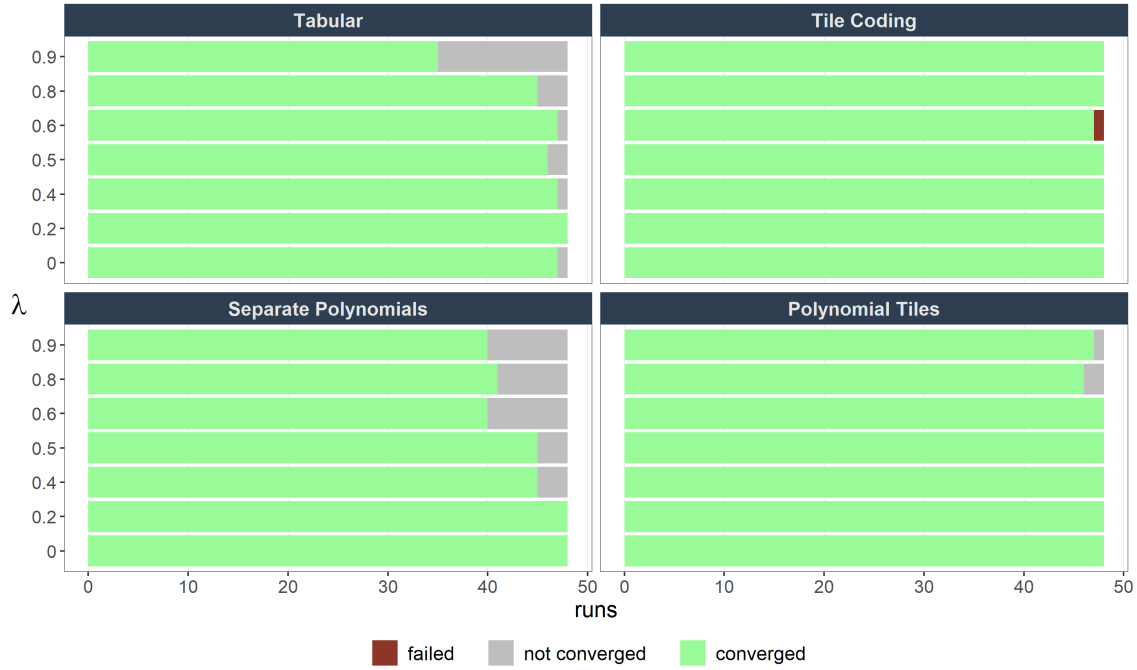

Figure 22: Number of runs per experiments that (i) achieved convergence, (ii) did not converge or (iii) failed to complete as a function of FEM and λ.

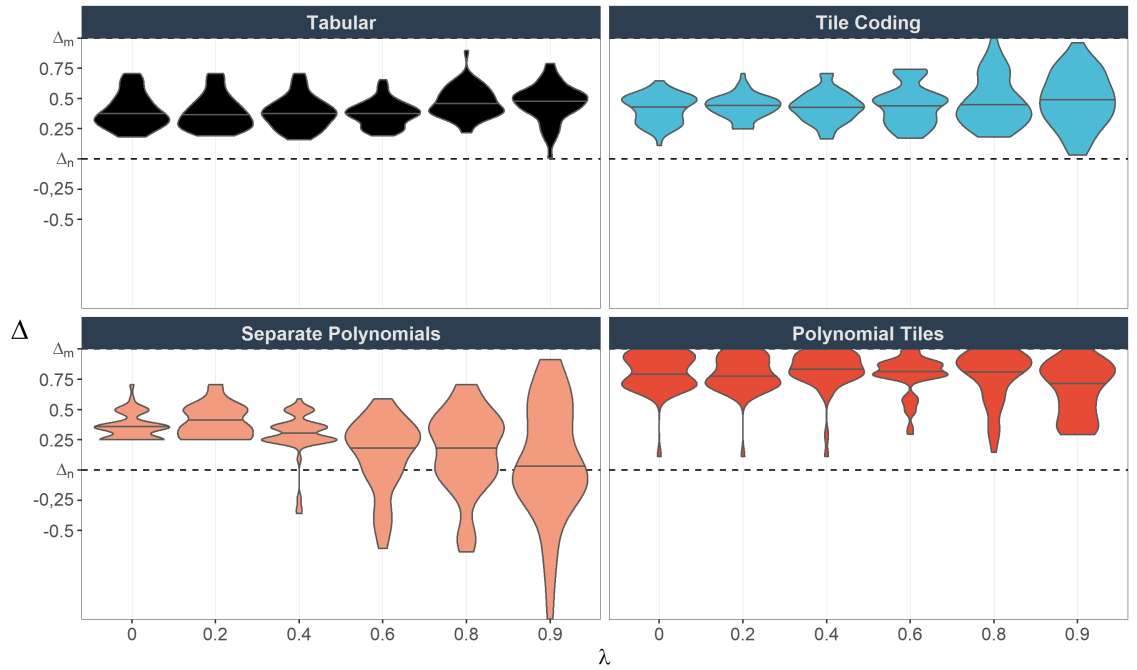

Figure 23: Distribution of Δ by FEM and λ. Includes converged and non-converged runs. Violin widths are scaled to maximize width of individual violins, comparisons of widths between violins are not meaningful. Violins are trimmed at smallest and largest observation respectively.

C.3 Price grid

Recall that m determines the number of feasible prices and therefore increases the complexity of the learning task. This holds particularly true for tabular learning and, to a lesser extent, for separate polynomials (refer back to Table 2). Against that backdrop, it is unsurprising that Figure 24 shows less runs converging if m increases. The effect is most pronounced for tabular learning and tile coding. Figure 25 shows the average response to a deviation for $m = 10$. Compared to the baseline parametrization, the punishment of the cheated agent seems more severe with tabular learning and polynomial tiles. However, the low sample size of runs (16 per experiment) warrants cautious interpretation.

Next, I will consider variations in ζ. Recall from (5) that ζ controls the available excess range above the fully collusive price p_m. These prices are inferior to p_m in almost any situation and the simulations confirm that few runs converge in *supra-monopoly* prices. So how may a change in ζ affect the learning behavior? Most importantly, large values of ζ increase the share of available prices above p_m and decreases the share of *viable* prices within the range of p_m and p_n. Consequently, the agents may quickly discard a larger share of actions engendering low (or negative) rewards and *narrow down* the range of reasonable actions between p_n and p_m. Then, with fewer available actions, the optimization within that range might be facilitated. This might be particularly important with the separate polynomials FEM because agents could learn that certain polynomials associated with actions above p_m (or below p_n) consistently yield low rewards - irrespective of the preceding state set, refrain from playing them early in the simulation and focus on refining the polynomials of actions within the range of p_n and p_m.

There is an additional effect on both tiling methods. The thresholds of all tiles derive from the size of the action space. Therefore, tiles are resized and relocated. The natural consequence is that some state-actions combinations will be associated with different tiles. *A priori*, the effect on outcomes is hard to predict.

I conducted three additional experiments with $\zeta \in \{0.1, 0.5, 1.5\}$ to assess the impact of varying ζ while keeping m constant at 19 to ensure comparability between experiments.[50] Figure 26 illustrates that ζ significantly influences profits upon convergence. Across FEMs, the average Δ increases with ζ. This trend is most pronounced with polynomial tiles. To reiterate, prices close to the collusive solution are not necessarily evidence of a stable equilibrium with a *reward-punishment* scheme. If anything, the simulation runs in this study have suggested the opposite and it turns out that despite the differences in Δ, the

[50] Note however, other ζ may prohibit playing actions very close to p_n or p_m. For instance, with $\zeta = 1.5$, the price closest to $p_n = 1.473$ ($p_m = 1.925$) is 1.454 (1.908). These gaps are quite a bit higher than in the default specification.

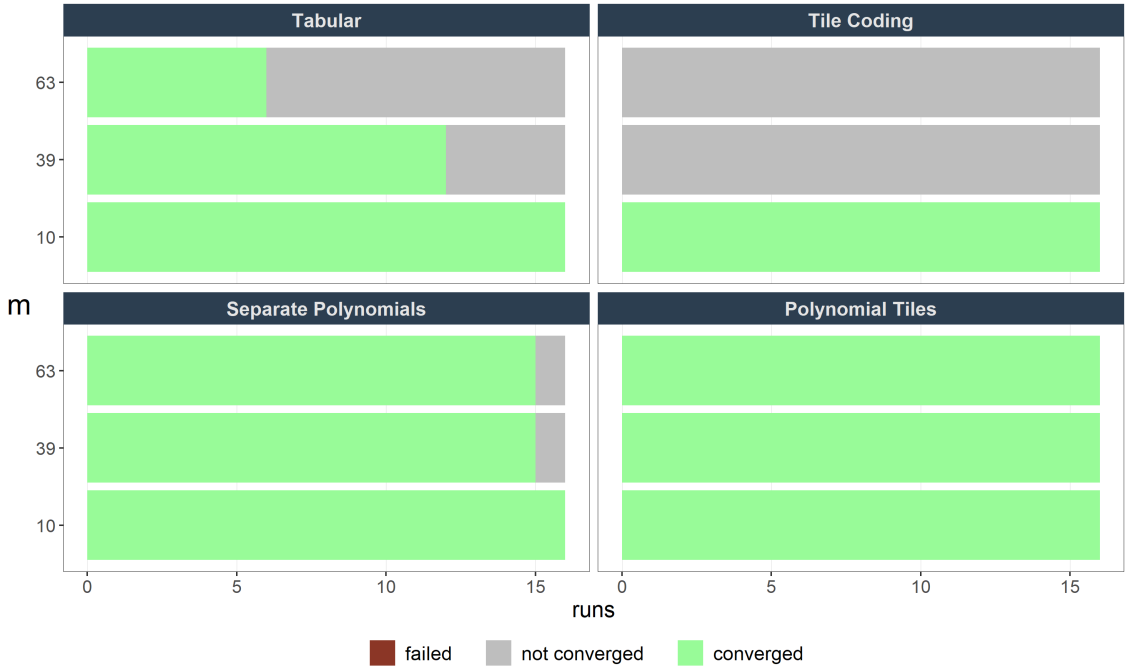

Figure 24: Number of runs per experiments that (i) achieved convergence, (ii) did not converge or (iii) failed to complete as a function of FEM and m. $m = 19$ is not plotted because the number of runs is not comparable (refer back to Figure 1).

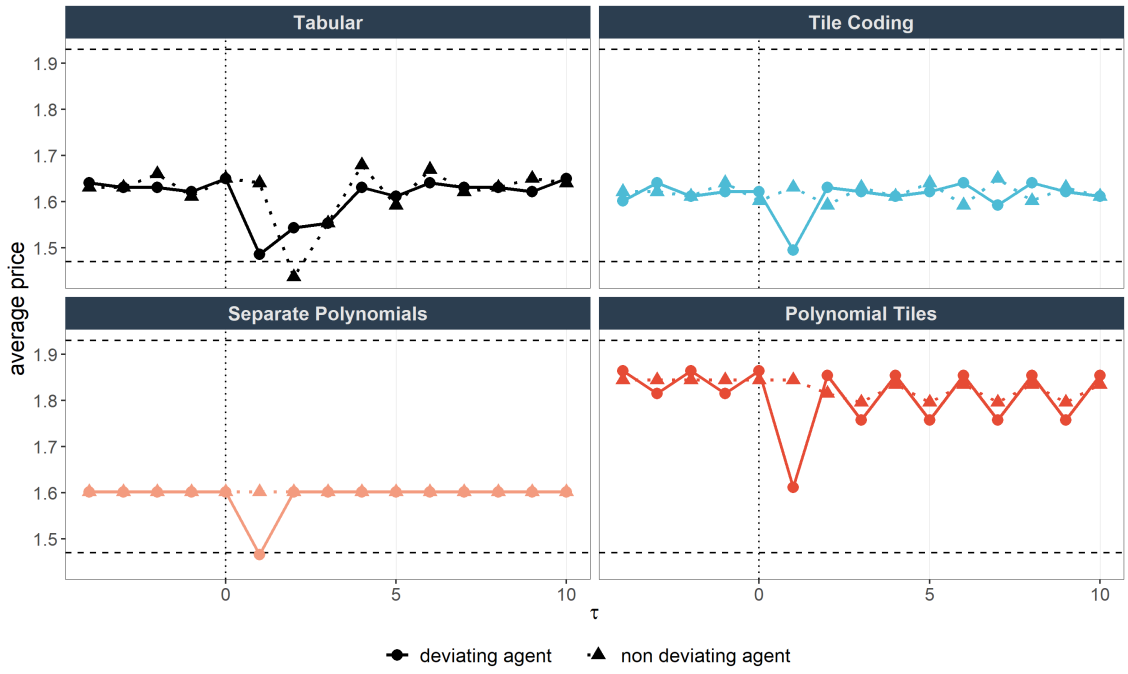

Figure 25: Average price trajectory around deviation by FEM with $m = 10$. Points represent the average price over all runs of an experiment. Dashed horizontal lines represent the fully collusive price p_m and the static Nash solution p_n. Dotted vertical line reflects time of convergence, i.e. the period immediately before the forced deviation.

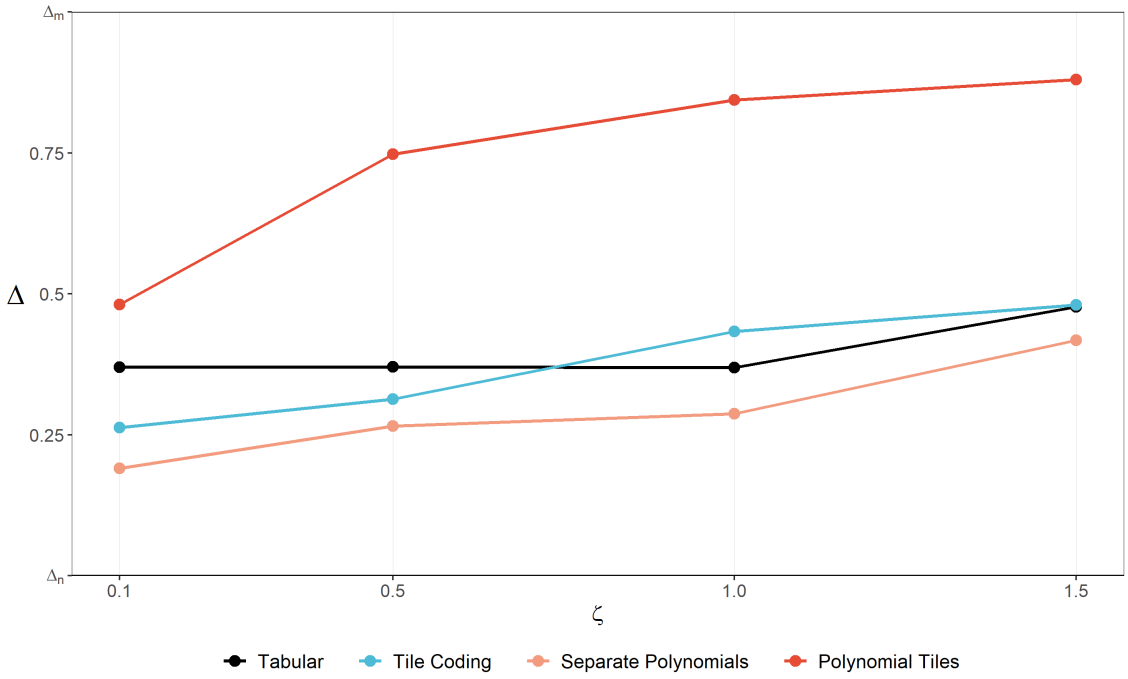

Figure 26: Average Δ by FEM and ζ. Includes converged and non-converged runs.

stability of the learned strategies is not heavily influenced by ζ.

Figure 27 confirms that average prices upon convergence largely remain within the Nash and collusive benchmarks. Polynomial tiles constitute the only exception. With $\zeta = 1$, a significant share of runs displays prices above p_m. This finding further discredits the FEM as appropriate for the learning task. Figure 28 displays the price trajectory during the forced deviation episode for tabular learning. Retaliatory pricing is visible in all variations.

FEM	agent	$\zeta = 0.1$	$\zeta = 0.5$	$\zeta = 1.0$	$\zeta = 1.5$
Tabular	deviating	0.24	0.21	0.24	0.31
Tabular	non deviating	0.07	0.12	0.07	0.04
Tile Coding	deviating	0.38	0.42	0.56	0.44
Tile Coding	non deviating	0.04	0.02	0.04	0.08
Separate Polynomials	deviating	0.46	0.57	0.69	0.60
Separate Polynomials	non deviating	0.00	0.00	0.00	0.00
Polynomial Tiles	deviating	0.83	0.96	0.85	0.84
Polynomial Tiles	non deviating	0.10	0.08	0.04	0.09

Table 8: Share of profitable deviations by FEM, agent and ζ. Annotations from Table 5 apply.

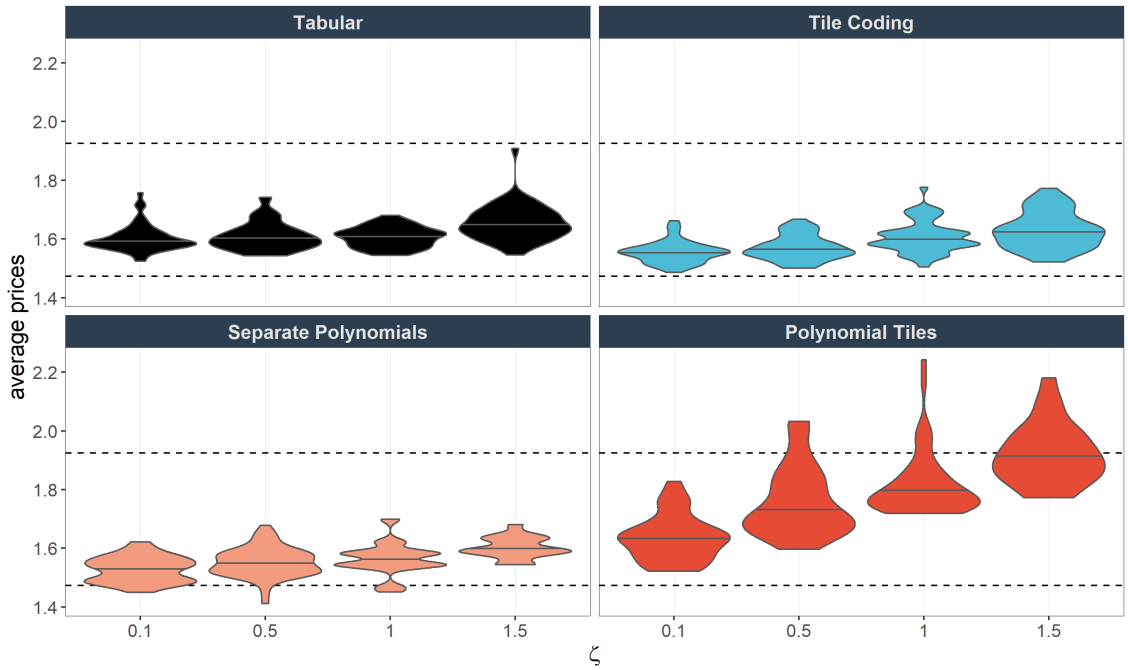

Figure 27: Distribution of average prices upon convergence by FEM and ζ. Includes converged and non-converged runs. Violin widths are scaled to maximize width of individual violins, comparisons of widths between violins are not meaningful. Violins are trimmed at smallest and largest observation respectively. Horizontal lines represent the median.

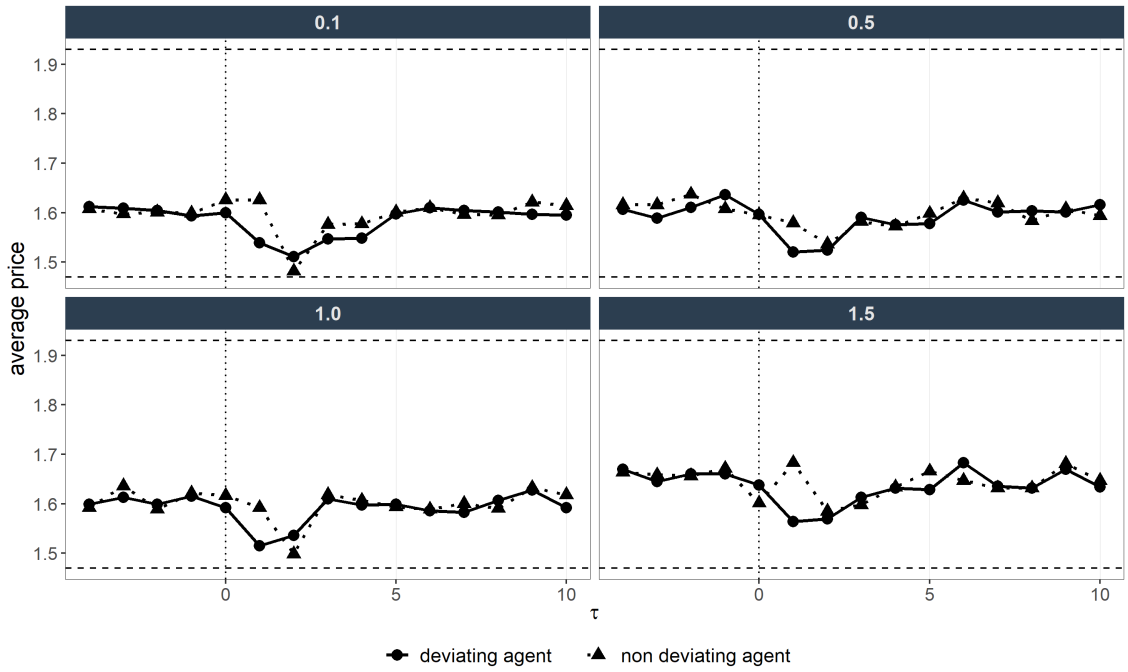

Figure 28: Average price trajectory around deviation by ζ. Numbers on strip represent values of ζ. Includes only tabular learning experiments. Points represent the average price over all runs of an experiment. Dashed horizontal lines represent the fully collusive price p_m and the static Nash solution p_n. Dotted vertical line reflects time of convergence, i.e. the period immediately before the forced deviation.

C.4 Differential reward setting

Section 4.3 described the *differential reward* setting, an alternative method to incorporate rewards into the learning process. I also described how the separated polynomial method struggled to achieve convergence in the alternative setting (refer back to Figure 8). Figure 29 emphasizes that point. A surprisingly large number of runs converges at a stage where exploration is incredibly rare. This suggests that agents, despite continuous exploitation, frequently change their evaluation of what the optimal action is.

Figure 30 displays for every experiment in the differential reward setting the range of Δ upon convergence. It appears that the considered values of υ do not impact the outcomes much. In comparison to the baseline runs, tabular learning and tile coding exhibit larger variation. In the case of tabular learning, some runs hover around Nash profits while others converge in equilibria close to the perfectly collusive benchmark.

Figure 31 illustrates the charged prices around the intervention relative to a counterfactual without a forced deviation for experiments with $\upsilon = 0.005$. Tabular learning shows a clear tendency to punish price cuts at $\tau = 2$. For tile coding and polynomial tiles, a price cut in response to the deviation occurs in *some* runs.

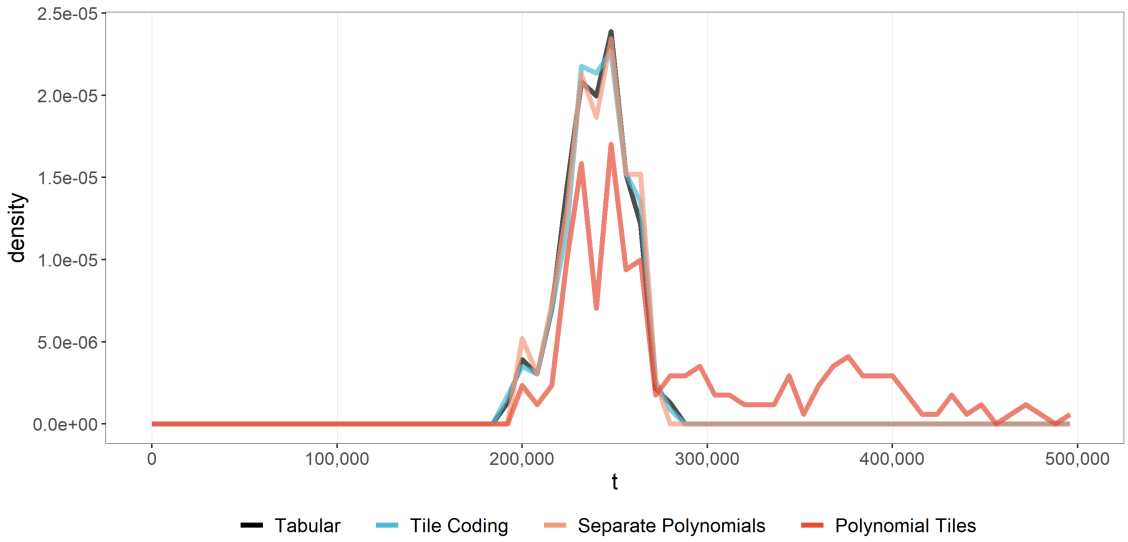

Figure 29: Timing of convergence in *differential reward* setting by FEM. only includes converged runs. Width of bins: 8,000.

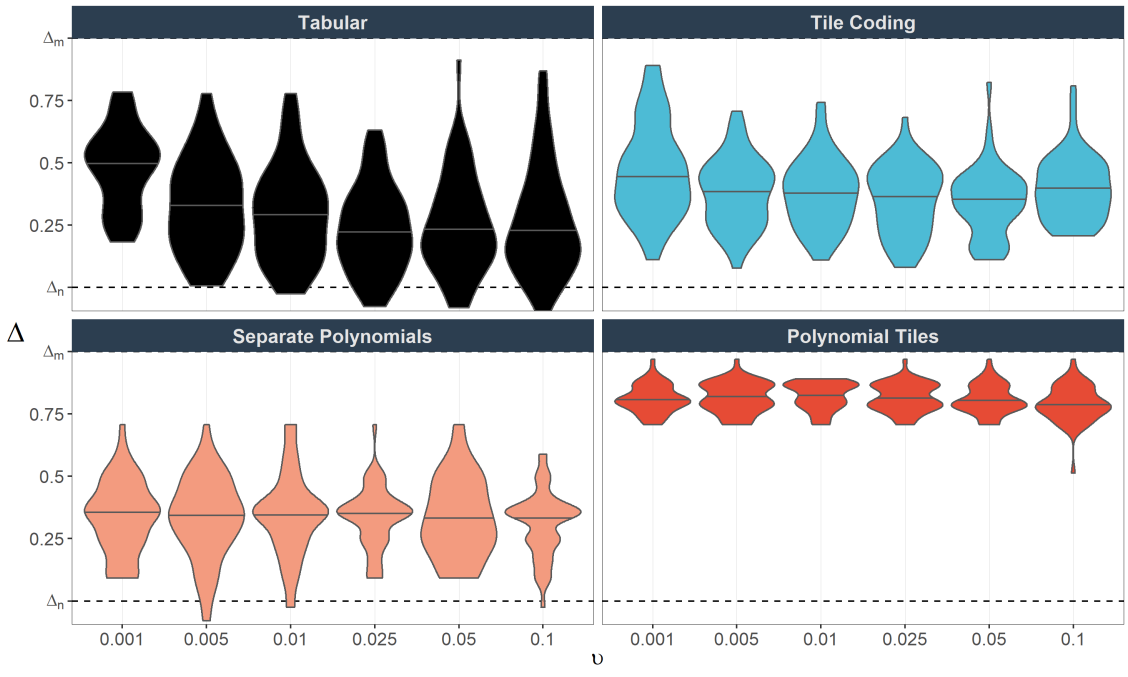

Figure 30: Distribution of Δ by FEM and v. Includes converged and non-converged runs from experiments employing the differential reward setting. Violin widths are scaled to maximize width of individual violins, comparisons of widths between violins are not meaningful. Violins are trimmed at smallest and largest observation respectively.

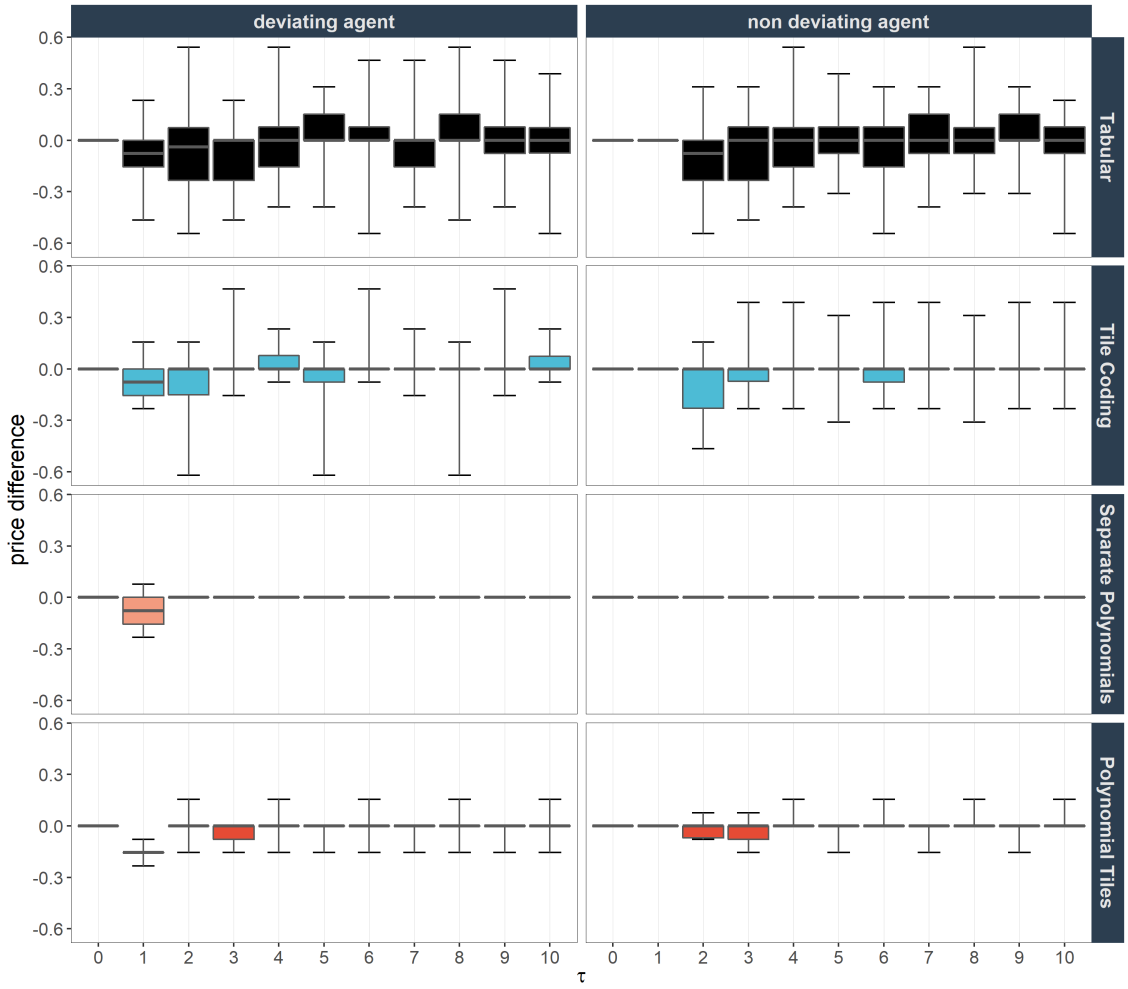

Figure 31: Distribution of price differences around deviation by FEM relative to counterfactual path *without* forced deviation, i.e. the difference to the price had no deviation taken place in the differential reward setting with $\upsilon = 0.005$. Only includes converged runs because a clear counterfactual exists. Boxes demarcate 15th and 85th percentiles. They are extended by whiskers that mark the entire range of price differences. Horizontal lines represent the group median.

C.5 Discount factor

In dynamic oligopolies, theory ascribes great importance to the discount factor γ. Typically, there exists a critical value below which the weight on future profits becomes too low to sustain any collusive behavior. Likewise, if γ is sufficiently high, rational actors with full information will collude on the monopoly solution. In reality, there are various reasons why decision makers may end up charging prices between both extremes. For instance, they might not be fully aware of what exactly the benchmark prices are and might struggle to communicate and agree on a joint action (explicitly or tacitly). Similarly, in reinforcement learning, it is unlikely that there exists a strict dichotomy between fully collusive and perfectly competitive agents. Indeed, the results so far suggests that many intermediate levels are realistic. Nevertheless, with lower values of γ, less weight is put on the (expected) value of the future state in (13) and the immediate reward R_t gains relative importance. Accordingly, one would expect the agents to gradually approach the Nash benchmark as γ decreases.

To gauge the actual effect of γ on outcomes, I conducted a series of experiments ranging from perfectly myopic ($\gamma = 0$) to almost infinitely patient ($\gamma = 0.99$) agents.[51] Figure 32 summarizes the variation in average Δ. Though the relationship is not as clear as anticipated, the curves of tabular learning and tile coding confirm the hypothesized pattern. With $\gamma = 0$, the average profits are much closer to the Nash benchmark. Another interesting revelation is that the *average* profits for tabular learning are highest at $\gamma = 0.85$ (average $\Delta = 0.48$). This suggests the agents struggle with high variance if γ approaches 1 (Naik et al. 2019).

With regard to the polynomial FEMs, the figure serves as further evidence of their ineptness for the considered learning task. Even without discounting ($\gamma = 0$), the outcomes remain high. In fact, they are even higher with separate polynomials. This clearly hints at a failure to learn how to compete when *only* the immediate reward should matter.[52]

With regard to prices, the expectation is that agents price closer to the competitive benchmark p_n in order to increase immediate profits. Indeed, Figure 33 shows that the distribution of average prices clearly shifts downwards for three of the four FEMs. The

[51] While $\gamma = 1$ is usually easy to model in economics, it is highly problematic in continuing learning tasks due to its infinite sum property (this is the main reason why discounting is commonly utilized in reinforcement learning in the first place, see e.g. Schwartz (1993)). Consider the following example. An agent with no time preference ($\gamma = 1$) in a continuous task explores early that a particular action consistently yields positive rewards. When *exploiting*, the agent keeps playing that action and the value estimate accumulates to infinity. This results in a significant bias towards actions that have been explored early and at some point becomes computationally infeasible. Through similar reasoning, values marginally below 1 are known to be unstable (Naik et al. 2019).

[52] I interpret this similar to the results in Waltman and Kaymak (2008) where *memoryless* agents without the ability to assert whether the opponent cheated still learn to charge supra-competitive prices.

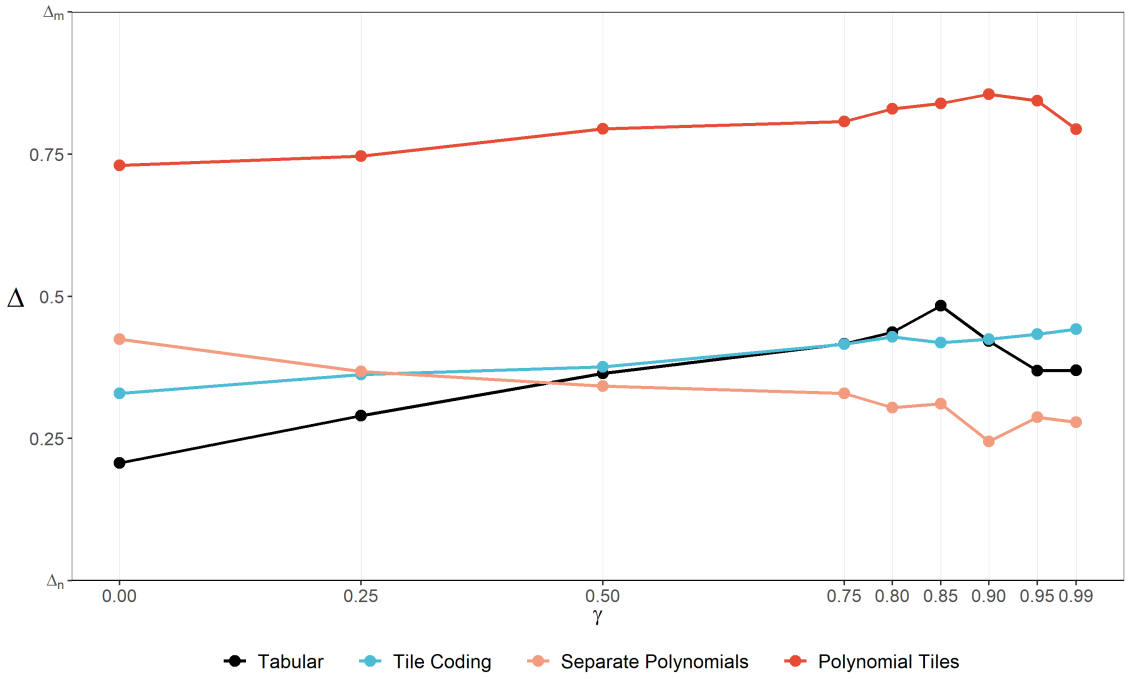

Figure 32: Average Δ by FEM and γ. Includes converged and non-converged runs.

effect is very clear for tabular learning and polynomial tiles. With regard to the latter, note that prices are still far above p_n. This is puzzling. Without regard for future profits one would expect agents to end up very close to the Nash solution. Likewise, γ's effect on average prices with tile coding points in the expected direction but is unexpectedly subtle. With separate polynomials, the impact of γ is not obvious. If anything, the plot suggests that low discount factors increase profits.

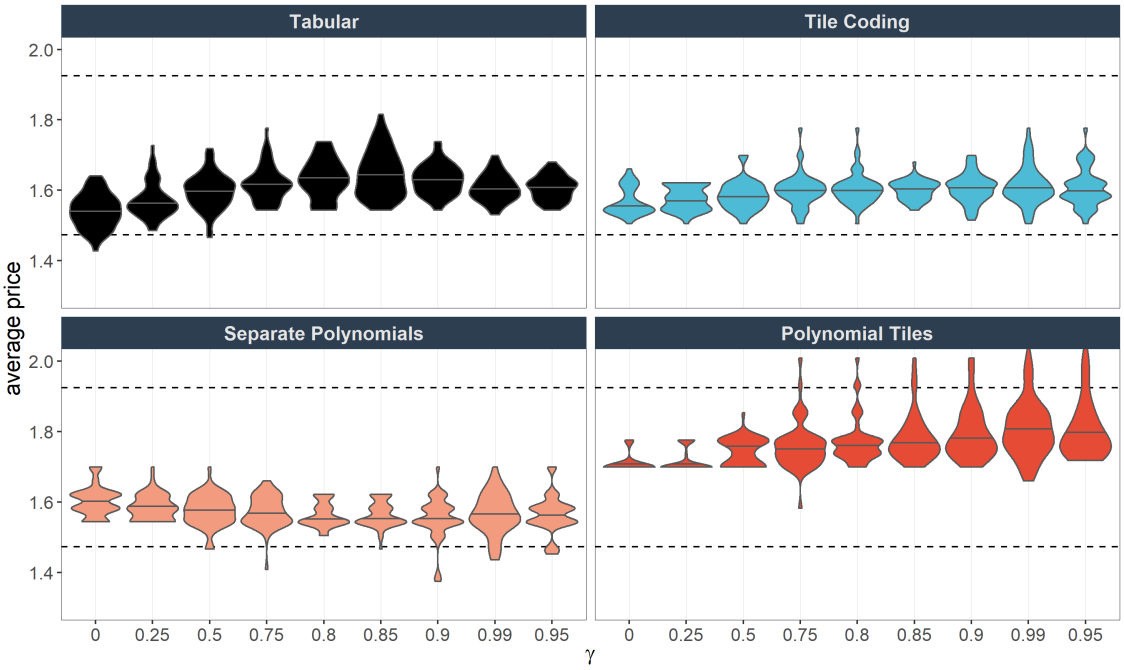

Figure 33: Distribution of average prices upon convergence by FEM and γ. Includes converged and non-converged runs. Violin widths are scaled to maximize width of individual violins, comparisons of widths between violins are not meaningful. Violins are trimmed at smallest and largest observation respectively. Horizontal lines represent the median.

C.6 Alternative algorithms

Of course, the specific algorithm described in Algorithm 1 is only one of many ways to use function approximation in learning tasks. I will consider two variations: *Tree backup* and *on-policy SARSA*.

C.6.1 Tree backup

Precup, Sutton, and Singh (2000) suggest the *tree backup* algorithm as a successor to Q-Learning. Compared to the *expected SARSA* algorithm, the update in (14) is replaced by:

$$\boldsymbol{z}_t = \gamma\lambda\kappa(A_t|S_t)\boldsymbol{z}_{t-1} + \frac{\Delta\hat{q}}{\Delta\boldsymbol{w}_t} \tag{29}$$

Recall that $\kappa(A_t|S_t)$ represents the probability of choosing A_t if the agent were to follow a hypothetical target policy with $\epsilon = 0$. As with the eligibility trace in expected SARSA, the idea is that $\boldsymbol{z}$ resets to $\mathbf{0}$ as soon as a non-greedy action is played. Unsurprisingly, applying the tree backup algorithm with optimized values of α to the environment yields not very different results. Figure 34 displays the distribution of Δ which is reminiscent of the violins for optimized values of α in Figure 12.

Similarly, the deviation experiments do not reveal new insights either. Figure 35 displays the average price trajectory around the deviation episode due to runs utilizing the *tree-backup* algorithm. The panels reiterate that only tabular learning agents show a consistent punishment in response to the forced deviation and the cheated agents learning through function approximation FEMs fail to respond in a compelling way. The bottom right panel, representing the polynomial tiles FEM, hints at a vague *matching strategy* culminating in new equilibria. But averaging turns out to be deceptive here. In fact, only in 12.5% of the runs does the non deviating agent respond with a price cut. Figure 36 displays the distribution of prices around the forced deviation. With regard to polynomial tiles, *some* runs show a sort of punishment or matching behavior in the wake of a price cut, but the vast majority (87.5%) of runs show no response.

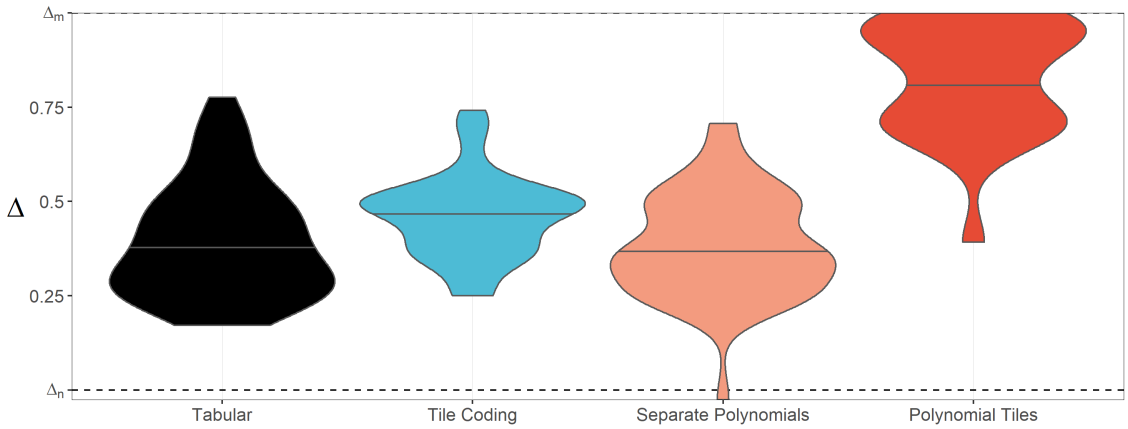

Figure 34: Distribution of Δ by FEM with *tree backup* algorithm. Includes converged and non-converged runs. Violin widths are scaled to maximize width of individual violins, comparisons of widths between violins are not meaningful. Violins are trimmed at smallest and largest observation respectively. Horizontal lines represent the median.

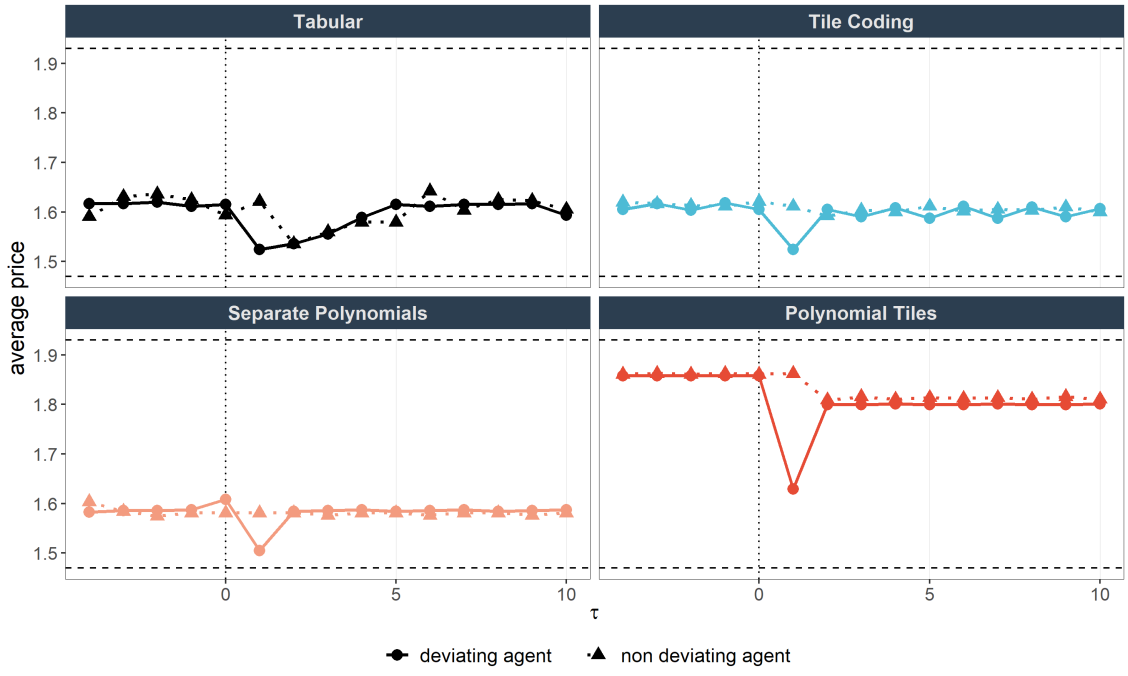

Figure 35: Average price trajectory around deviation by FEM with *tree backup* algorithm. Points represent the average price over all runs of an experiment. Dashed horizontal lines represent the fully collusive price p_m and the static Nash solution p_n. Dotted vertical line reflects time of convergence, i.e. the period immediately before the forced deviation.

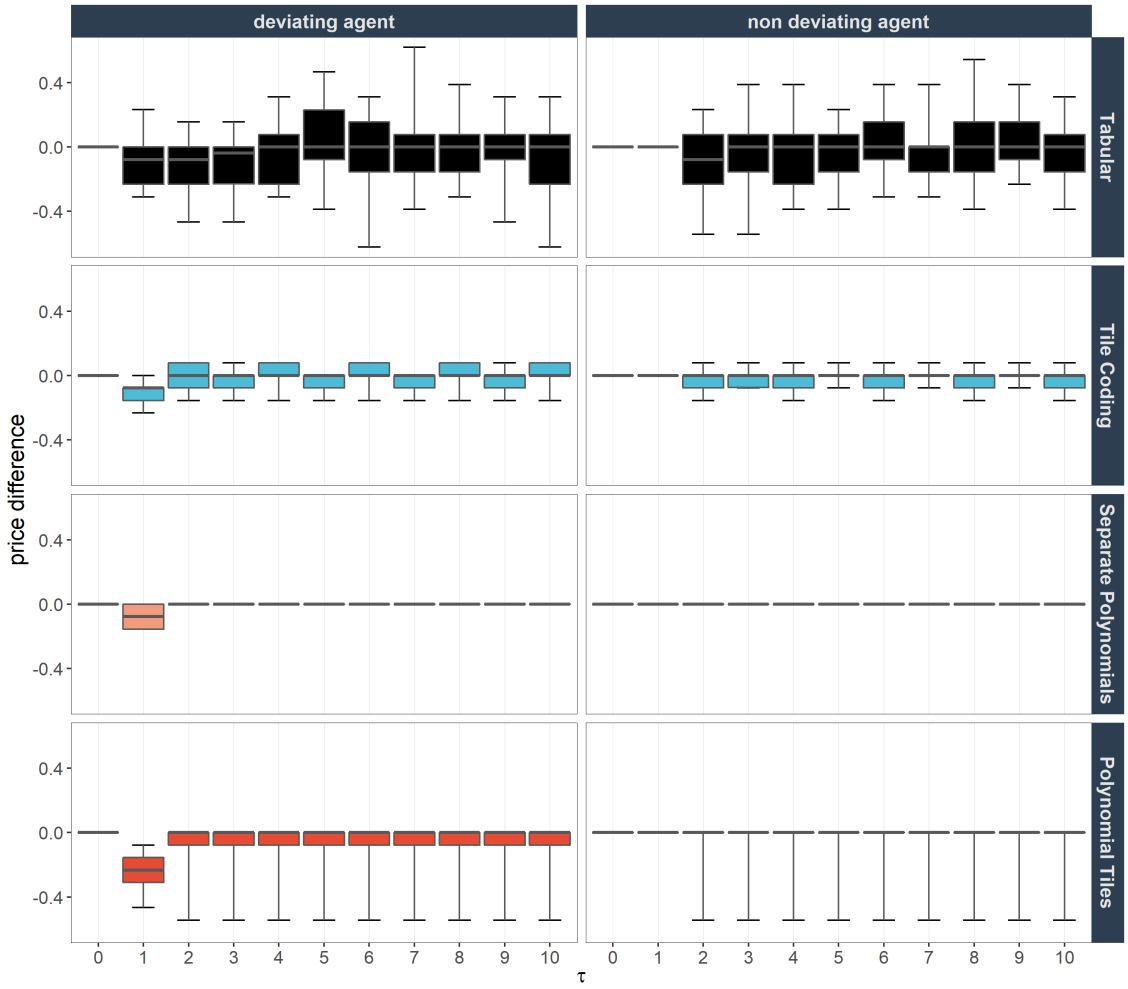

Figure 36: Distribution of price differences around deviation relative to counterfactual path *without* forced deviation, i.e. the difference to the price had no deviation taken place by FEM with *tree backup* algorithm. Only includes converged runs because a clear counterfactual exists. Boxes demarcate 15th and 85th percentiles. They are extended by whiskers that mark the entire range of price differences. Horizontal lines represent the group median.

C.6.2 On-policy SARSA

Q-Learning, *tree backup* and *expected SARSA* all belong to the family of *off-policy* learning algorithms. This stems from the simple fact that the (discounted) value estimation of the state-action combination at $t+1$ is not always based on the actually chosen action A_{t+1}.[53] So, it is *off-path* of the actually pursued policy. Off-policy methods tend to exhaust the entire range of state-action combination well, but convergence guarantees for them are generally weaker than for *on-policy* algorithms (Sutton and Barto 2018).[54] As their name suggests, *on-policy* algorithms wait until the state-action combination at $t+1$ is actually known and only then estimate the TD error δ_t. A straightforward adaption is:

$$\delta_t^{SARSA} = r_t + \gamma \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t) \quad , \tag{30}$$

Note that learning is delayed in the sense that δ_t^{SARSA} can only be calculated after the action in the next period has been taken. Algorithm 2 documents the *on-policy* algorithm in all its steps. Note that ρ does not appear in the update system anymore. This is precisely because the algorithm only takes into account the actually selected action at $t+1$.

Algorithm 2 SARSA (on policy)

input feasible prices via $m \in \mathbb{N}$ and $\zeta > 0$
configure static algorithm parameters $\alpha > 0$, $\beta > 0$, and $\lambda \in [0, 1]$
initialize parameter vector and eligibility trace $\boldsymbol{w} = \boldsymbol{z} = \boldsymbol{0}$
declare convergence rule (see section 3.1)
start tracking time: $t = 1$
randomly initialize state S_t
choose initial action A_t
while convergence is not achieved, **do**
  observe profit π, adjust to reward r
  move to next state: $t \leftarrow t+1$ and $S_{t+1} \leftarrow A_t$
  select action A_{t+1} according to (11)
  calculate TD-error (30): $\delta \leftarrow r + \gamma \hat{q}(S_{t+1}, A_{t+1}) - \hat{q}(S_t, A_t)$
  update eligibility trace: $\boldsymbol{z} \leftarrow \gamma\lambda\boldsymbol{z} + \boldsymbol{x}$
  update parameter vector (15): $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\delta\boldsymbol{z}$
  $S \leftarrow S_{t+1}$ and $A \leftarrow A_{t+1}$
end while

Using the optimized values of α, I conducted one experiment per FEM. Figure 37 illustrates that the distribution of outcomes per experiment resembles the two *off-policy* algorithms. Overall, the conclusions drawn in the previous section also apply to the *on-policy* algorithm.

[53] See (13). The only exception is $\epsilon = 0$.

[54] The main reason why I haven't put much consideration into this is that due to the *moving target problem* described in section 2.4, convergence is not guaranteed anyway. Moreover, Hettich (2021) shows that off-policy methods can work well with function approximation.

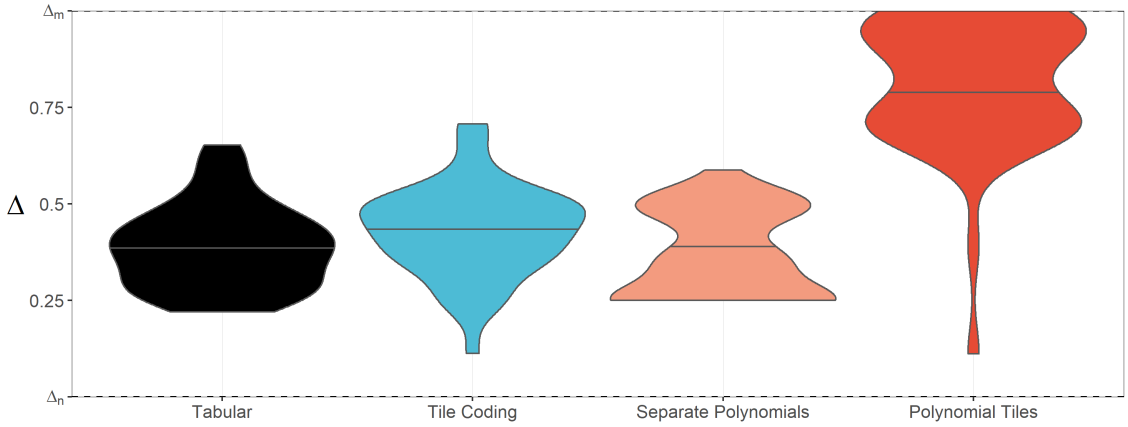

Figure 37: Distribution of Δ by FEM with *on-policy* algorithm . Includes converged and non-converged runs. Violin widths are scaled to maximize width of individual violins, comparisons of widths between violins are not meaningful. Violins are trimmed at smallest and largest observation respectively. Horizontal lines represent the median.

PREVIOUS DISCUSSION PAPERS

370 Jeschonneck, Malte, Collusion among Autonomous Pricing Algorithms Utilizing Function Approximation Methods, August 2021.

369 Gösser, Niklas, Gürer, Kaan, Haucap, Justus, Meyring, Bernd, Michailidou, Asimina, Schallbruch, Martin, Seeliger, Daniela and Thorwarth, Susanne, Total Consumer Time – A New Approach to Identifying Digital Gatekeepers, August 2021.

368 Fischer, Kai, Reade, J. James and Schmal, W. Benedikt, The Long Shadow of an Infection: COVID-19 and Performance at Work, August 2021.

367 Suedekum, Jens, Place-Based Policies – How to Do Them and Why, August 2021.

366 Heiss, Florian, Ornaghi, Carmine and Tonin, Mirco, Inattention vs Switching Costs: An Analysis of Consumers' Inaction in Choosing a Water Tariff, July 2021.

365 Cobb-Clark, Deborah A., Dahmann, Sarah C., Kamhöfer, Daniel A. and Schildberg-Hörisch, Hannah, Sophistication about Self-Control, July 2021.

364 Bie, Xiaodong and Ciani, Andrea, Born Similar, Develop Apart: Evidence on Chinese Hybrid Exporters, July 2021.

363 Ali, Nesma and Stiebale, Joel, Foreign Direct Investment, Prices and Efficiency: Evidence from India, July 2021.

362 Banerjee, Ritwik, Ibanez, Marcela, Riener, Gerhard and Sahoo, Soham, Affirmative Action and Application Strategies: Evidence from Field Experiments in Columbia, April 2021.

361 Wellmann, Nicolas and Czarnowske, Daniel, What Would Households Pay for a Reduction of Automobile Traffic? Evidence From Nine German Cities, March 2021.

360 Haucap, Justus, Moshgbar, Nima and Schmal, Wolfgang Benedikt, The Impact of the German "DEAL" on Competition in the Academic Publishing Market, March 2021.

359 Korff, Alex, Competition in the Fast Lane – The Price Structure of Homogeneous Retail Gasoline Stations, January 2021.

358 Kiessling, Lukas, Chowdhury, Shyamal, Schildberg-Hörisch, Hannah and Sutter, Matthias, Parental Paternalism and Patience, January 2021.

357 Kellner, Christian, Le Quement, Mark T. and Riener, Gerhard, Reacting to Ambiguous Messages: An Experimental Analysis, December 2020.

356 Petrishcheva,Vasilisa, Riener, Gerhard and Schildberg-Hörisch, Hannah, Loss Aversion in Social Image Concerns, November 2020.

355 Garcia-Vega, Maria, Kneller, Richard and Stiebale, Joel, Labor Market Reform and Innovation: Evidence from Spain, November 2020.
Published in: Research Policy, 50 (2021), 104213.

354 Steffen, Nico, Economic Preferences, Trade and Institutions, November 2020.

353 Pennerstorfer, Dieter, Schindler, Nora, Weiss, Christoph and Yontcheva, Biliana, Income Inequality and Product Variety: Empirical Evidence, October 2020.

352 Gupta, Apoorva, R&D and Firm Resilience During Bad Times, October 2020.

351 Shekhar, Shiva and Thomes, Tim Paul, Passive Backward Acquisitions and Downstream Collusion, October 2020.
Forthcoming in: Economics Letters.

350 Martin, Simon, Market Transparency and Consumer Search – Evidence from the German Retail Gasoline Market, September 2020.

349 Fischer, Kai and Haucap, Justus, Betting Market Efficiency in the Presence of Unfamiliar Shocks: The Case of Ghost Games during the COVID-19 Pandemic, August 2020.

348 Bernhardt, Lea, Dewenter, Ralf and Thomas, Tobias, Watchdog or Loyal Servant? Political Media Bias in US Newscasts, August 2020.

347 Stiebale, Joel, Suedekum, Jens and Woessner, Nicole, Robots and the Rise of European Superstar Firms, July 2020.

346 Horst, Maximilian, Neyer, Ulrike and Stempel, Daniel, Asymmetric Macroeconomic Effects of QE-Induced Increases in Excess Reserves in a Monetary Union, July 2020.

345 Riener, Gerhard, Schneider, Sebastian O. and Wagner, Valentin, Addressing Validity and Generalizability Concerns in Field Experiments, July 2020.

344 Fischer, Kai and Haucap, Justus, Does Crowd Support Drive the Home Advantage in Professional Soccer? Evidence from German Ghost Games during the COVID-19 Pandemic, July 2020.
Forthcoming in: Journal of Sports Economics.

343 Gösser, Niklas and Moshgbar, Nima, Smoothing Time Fixed Effects, July 2020.

342 Breitkopf, Laura, Chowdhury, Shyamal, Priyam, Shambhavi, Schildberg-Hörisch, Hannah and Sutter, Matthias, Do Economic Preferences of Children Predict Behavior?, June 2020.

341 Westphal, Matthias, Kamhöfer, Daniel A. and Schmitz, Hendrik, Marginal College Wage Premiums under Selection into Employment, June 2020.

340 Gibbon, Alexandra J. and Schain, Jan Philip, Rising Markups, Common Ownership, and Technological Capacities, April 2021 (First Version June 2020).

339 Falk, Armin, Kosse, Fabian, Schildberg-Hörisch, Hannah and Zimmermann, Florian, Self-Assessment: The Role of the Social Environment, May 2020.

338 Schildberg-Hörisch, Hannah, Trieu, Chi and Willrodt, Jana, Perceived Fairness and Consequences of Affirmative Action Policies, April 2020.

337 Avdic, Daniel, de New, Sonja C. and Kamhöfer, Daniel A., Economic Downturns and Mental Wellbeing, April 2020.

336 Dertwinkel-Kalt, Markus and Wey, Christian, Third-Degree Price Discrimination in Oligopoly When Markets Are Covered, April 2020.

335 Dertwinkel-Kalt, Markus and Köster, Mats, Attention to Online Sales: The Role of Brand Image Concerns, April 2020.
Forthcoming in: Journal of Economics and Management Strategy

334 Fourberg, Niklas and Korff, Alex, Fiber vs. Vectoring: Limiting Technology Choices in Broadband Expansion, April 2020.
Published in: Telecommunications Policy, 44 (2020), 102002.

333 Dertwinkel-Kalt, Markus, Köster, Mats and Sutter, Matthias, To Buy or Not to Buy? Price Salience in an Online Shopping Field Experiment, April 2020.
Revised version published in: European Economic Review, 130 (2020), 103593.

332 Fischer, Christian, Optimal Payment Contracts in Trade Relationships, February 2020.

331 Becker, Raphael N. and Henkel, Marcel, The Role of Key Regions in Spatial Development, February 2020.

330 Rösner, Anja, Haucap, Justus and Heimeshoff, Ulrich, The Impact of Consumer Protection in the Digital Age: Evidence from the European Union, January 2020.
Published in: International Journal of Industrial Organization, 73 (2020), 102585.

329 Dertwinkel-Kalt, Markus and Wey, Christian, Multi-Product Bargaining, Bundling, and Buyer Power, December 2019.
Published in: Economics Letters, 188 (2020), 108936.

328 Aghelmaleki, Hedieh, Bachmann, Ronald and Stiebale, Joel, The China Shock, Employment Protection, and European Jobs, December 2019.

327 Link, Thomas, Optimal Timing of Calling In Large-Denomination Banknotes under Natural Rate Uncertainty, November 2019.

326 Heiss, Florian, Hetzenecker, Stephan and Osterhaus, Maximilian, Nonparametric Estimation of the Random Coefficients Model: An Elastic Net Approach, September 2019.
Forthcoming in: Journal of Econometrics.

325 Horst, Maximilian and Neyer, Ulrike, The Impact of Quantitative Easing on Bank Loan Supply and Monetary Policy Implementation in the Euro Area, September 2019.
Published in: Review of Economics, 70 (2019), pp. 229-265.

324 Neyer, Ulrike and Stempel, Daniel, Macroeconomic Effects of Gender Discrimination, September 2019.

323 Stiebale, Joel and Szücs, Florian, Mergers and Market Power: Evidence from Rivals' Responses in European Markets, September 2019.

322 Henkel, Marcel, Seidel, Tobias and Suedekum, Jens, Fiscal Transfers in the Spatial Economy, September 2019.
Forthcoming in: American Economic Journal: Economic Policy.

321 Korff, Alex and Steffen, Nico, Economic Preferences and Trade Outcomes, August 2019.

320 Kohler, Wilhelm and Wrona, Jens, Trade in Tasks: Revisiting the Wage and Employment Effects of Offshoring, July 2019.
Forthcoming in: Canadian Journal of Economics.

319 Cobb-Clark, Deborah A., Dahmann, Sarah C., Kamhöfer, Daniel A. and Schildberg-Hörisch, Hannah, Self-Control: Determinants, Life Outcomes and Intergenerational Implications, July 2019.

318 Jeitschko, Thomas D., Withers, John A., Dynamic Regulation Revisited: Signal Dampening, Experimentation and the Ratchet Effect, July 2019.

317 Jeitschko, Thomas D., Kim, Soo Jin and Yankelevich, Aleksandr, Zero-Rating and Vertical Content Foreclosure, July 2019.
Published in: Information Economics and Policy, 55 (2021), 100899.

316 Kamhöfer, Daniel A. und Westphal, Matthias, Fertility Effects of College Education: Evidence from the German Educational Expansion, July 2019.

315 Bodnar, Olivia, Fremerey, Melinda, Normann, Hans-Theo and Schad, Jannika, The Effects of Private Damage Claims on Cartel Activity: Experimental Evidence, June 2021 (First Version June 2019 under the title "The Effects of Private Damage Claims on Cartel Stability: Experimental Evidence").
Forthcoming in: Journal of Law, Economics, and Organization.

314 Baumann, Florian and Rasch, Alexander, Injunctions Against False Advertising, October 2019 (First Version June 2019).
Published in: Canadian Journal of Economics, 53 (2020), pp. 1211-1245.

313 Hunold, Matthias and Muthers, Johannes, Spatial Competition and Price Discrimination with Capacity Constraints, May 2019 (First Version June 2017 under the title "Capacity Constraints, Price Discrimination, Inefficient Competition and Subcontracting").
Published in: International Journal of Industrial Organization, 67 (2019), 102524.

312 Creane, Anthony, Jeitschko, Thomas D. and Sim, Kyoungbo, Welfare Effects of Certification under Latent Adverse Selection, March 2019.

311 Bataille, Marc, Bodnar, Olivia, Alexander Steinmetz and Thorwarth, Susanne, Screening Instruments for Monitoring Market Power – The Return on Withholding Capacity Index (RWC), March 2019.
Published in: Energy Economics, 81 (2019), pp. 227-237.

310 Dertwinkel-Kalt, Markus and Köster, Mats, Salience and Skewness Preferences, March 2019.
Published in: Journal of the European Economic Association, 18 (2020), pp. 2057–2107.

309 Hunold, Matthias and Schlütter, Frank, Vertical Financial Interest and Corporate Influence, February 2019.

308 Sabatino, Lorien and Sapi, Geza, Online Privacy and Market Structure: Theory and Evidence, February 2019.

307 Izhak, Olena, Extra Costs of Integrity: Pharmacy Markups and Generic Substitution in Finland, January 2019.

Older discussion papers can be found online at:
http://ideas.repec.org/s/zbw/dicedp.html

Heinrich-Heine-Universität Düsseldorf

Düsseldorfer Institut für Wettbewerbsökonomie (DICE)

Universitätsstraße 1, 40225 Düsseldorf

ISSN 2190-992X (online)
ISBN 978-3-86304-369-8