Tacit Algorithmic Collusion

A Replication of Calvano et al. (*American Economic Review*, 2020)

Jacob Schildknecht*

Journal of Comments and Replications in Economics, Volume 5, 2026-7, DOI: 10.18718/81781.58

JEL: L41, L13, D43, D83
Keywords: Algorithmic collusion, Reinforcement learning, Competition policy, Pricing algorithms, Replication study
Data Availability: The Python code to reproduce the results of this replication can be downloaded at JCRE's data archive (DOI: 10.15456/j1.2026111.1137134085).

Please Cite As: Schildknecht, J. (2026). Tacit Algorithmic Collusion. A Replication of Calvano et al. (*American Economic Review*, 2020). *Journal of Comments and Replications in Economics*, Vol.5 (2026-7). DOI: 10.18718/81781.58

Abstract

The adoption of pricing algorithms in markets has raised concerns about their potential to collude, even without explicit programming or communication. This paper reproduces and validates a study by Calvano et al. (2020), demonstrating that reinforcement learning-based pricing algorithms can learn to collude tacitly. After translating their original Fortran implementation into Python, I confirm the result that algorithms are able to collude and achieve supra-competitive prices and profit gains of 70-90% in equilibrium. I also confirm that they learn sophisticated punishment strategies by going into a price war before returning to equilibrium. I provide the Python replication code as open source, making it easier for subsequent research to reuse it jointly with the large existing open source codebase for reinforcement learning and pricing algorithms. The findings underscore the concerns and highlight the need for future regulation of algorithmic collusion.

*ZEW Mannheim, jacob.schildknecht@zew.de

Received July 15, 2025; Revised January 5, 2026; Accepted January 8, 2026; Published April 21, 2026.

1 Introduction

Digital transformation changes the way firms compete in retail markets. Merchants across various sectors rely on pricing algorithms to monitor prices of competitors and recalibrate their prices in reaction. While this shift towards algorithmic pricing has the potential to enhance market efficiency, there is also a dark potential: algorithms that collude to maintain high prices, even in markets that have not been prone to collusive pricing behavior.

This prospect of autonomous algorithmic collusion poses a novel and critical challenge to regulation. This form of collusion "would defy current antitrust policy, which typically targets only explicit agreements among would-be competitors" (Calvano et al., 2020, p. 3268). The primary concern is that pricing algorithms "can develop their pricing strategies from scratch" learning to raise prices "even if they have not been specifically instructed to do so and even if they do not communicate with one another" (Calvano et al., 2020, p. 3268). Because this behavior does not "leave any trace whatever of concerted action", this type of collusion is exceptionally hard to prosecute under existing legal frameworks.

Calvano et al. (2020) call the verification of this collusion "a difficult question to answer, both empirically and theoretically" (Calvano et al., 2020, p. 3268). There are two main issues in this question: firstly, "Collusion is notoriously hard to detect from market outcomes" and secondly "firms typically do not disclose details of the pricing software they use". Given these two challenges, the original study by Calvano et al. (2020) provides a "proof-of-concept demonstration of algorithmic collusion" (Calvano et al., 2020, p. 3269), making the validation of their foundational results a matter of significant academic and policy importance.

A crucial aspect of this validation is distinguishing genuine collusion from simple market outcomes. As Calvano et al. (2020) emphasize, "the observation of supracompetitive prices is not, per se, genuine proof of collusion." True collusion "crucially involves 'a reward-punishment scheme designed to provide the incentives for firms to consistently price above the competitive level.'" Therefore, a key objective of this study is to verify not just the emergence of high prices, but the presence of these underlying enforcement mechanisms. Confirming the existence of "sophisticated punishment strategies" is vital evidence that the observed behavior is genuine collusion rather than a simple "failure to optimize." (Calvano et al., 2020, p. 3269)

Given its foundational nature, the Calvano et al. (2020) model has served as a baseline for subsequent studies, including investigations into algorithm responses to price shocks (Epivent & Lambin, 2024) and the robustness of collusive policies outside simulated environments (Eschenbaum et al., 2022). Consequently, several open-source implementations have emerged, including Lewis (2024) in Julia, and Python implementations by Courthoud (2021), Werner (2021), and yusei406 (2024).

This replication package distinguishes itself through three key contributions: accessibility, computational scalability, and generalized scope. First, unlike the Julia-based Lewis (2024), this framework offers a Python architecture compatible with the modern deep reinforcement learning ecosystem (e.g., PyTorch, JAX or Tensorflow). Second, regarding scalability, the framework improves upon Courthoud (2021) through vectorized updates and, unlike *qpricesim*, utilizes native Python *multipro-*

cessing rather than external cluster schedulers (e.g., OpenPBS). This enables efficient large-scale simulations (1,000+ sessions) on standard multi-core workstations. Finally, unlike *calvano-thesis* which is restricted to symmetric firms and episodic training, this framework introduces an automated solver for asymmetric Nash and Joint-Profit-Maximization benchmarks. This allows for the precise replication of asymmetric cost and quality scenarios (see Table 2), a feature absent in previous open-source baselines.

This paper confirms the key findings by Calvano et al. that algorithms are able to collude and sustain prices above the competitive level. Through the use of basic reinforcement algorithms in a simulated, controlled market environment, I confirm that these algorithms achieve profit gains of 70-90% and learn sophisticated punishment strategies to maintain the collusion. The reproduction confirms both the supra-competitive prices and the emergence of punishment strategies, where algorithms respond to changes in prices with retaliatory price cuts before returning to the collusive level.

Beyond the reproduction, the study makes two further contributions. Primarily, by the use of Python, a widely used programming language in economics and data science, the methodology by Calvano et al. is made more accessible to other researchers and authorities. This is of particular value given the regulatory focus on algorithmic collusion during the ongoing discussion surrounding the EU's AI Act. Secondly, the Python implementation provides a framework to use more sophisticated machine learning algorithms and market mechanisms, enabling future research on algorithmic collusion.

The subsequent paper is structured as follows: section 2 provides the methodology including the market environment and the algorithms. section 3 presents the main results, comparing them with the original results by Calvano et al. (2020). section 4 employs the framework to test asymmetric costs and qualities following the methodology of Calvano et al. (2020). section 5 discusses the implications of the successful reproduction for future research and the subsequent policy implications. section 6 concludes.

2 Methods

This study implements the investigation by Calvano et al. (2020) about algorithmic collusion in a simulated duopoly environment, keeping the same market structure while using Python instead of Fortran. The market is characterized by a logit demand structure, with five key parameters that regulate the competitive dynamics: (1) a outside good parameter a_0, (2) a quality index a_i per product, (3) a cost index c_i per product, (4) a product differentiation parameter μ and (5) the number of firms n competing in the environment. Each product is supplied by a different firm, as in Calvano et al. The demand $q_{i,t}$ with respect to the prices $p_{i,t}$ for product i in period t is defined as in Calvano et al.:

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

The profit $\pi_{i,t}$ of firm i in period t is characterized by the following formula:

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

c_i is the marginal cost for the product. Fixed costs are not included in the model, as in Calvano et al.

In line with the original study, an analysis of duopolistic competition with symmetric firms is conducted, with product quality a_i fixed at 2.0, costs c_i at 1.0, the outside good a_0 at 0 and the product differentiation μ at 0.25. These values are taken over from the approach by Calvano et al. to ensure direct compatibility of the results. Prices are discretized to 15 price levels spanning from Nash equilibrium to monopoly price levels. The learning agents employ tabular Q-Learning, where pricing strategies evolve through trial and error based on realized profits. The learning process is governed by three key parameters. The chosen parameters in brackets are in line with the implementation by Calvano et al.:

- Learning rate (α = 0.15): determines how quickly agents update strategies based on new information
- Exploration decay (β = 0.000004): controls the balance between exploration (testing new strategies) and exploitation (playing known profitable ones)
- Discount factor (δ = 0.95): reflects the weight placed on future versus current profits

The exploration decay parameter controls how quickly the algorithm transitions from random exploration of the action space to exploiting known profitable strategies. As the exploration rate β decays, the algorithms learn to converge to profitable pricing patterns. The convergence criterion is defined as stable strategies in 100,000 consecutive periods.[1]

To enable broader access and future research, the original Fortran implementation is translated to Python. The Python implementation maintains the fundamental economic environment while

[1] Calvano et al. (2020) define stable strategies as the action $a_{i,t}(s)$ at time t for agent i in state s staying unchanged for 100,000 consecutive periods.

updating the technical architecture. The Python implementation utilizes numpy arrays to ensure computational efficiency while preserving the model structure. Comprehensive implementation comparisons and detailed technical specifications are provided in Appendix Table 3. The Python implementation of this study is available at GitHub to facilitate future research on algorithmic collusion.

The analysis draws upon 1,000 independent training sessions using these baseline parameters. For each session, convergence to stable strategies, price trajectories and profit gains are tracked to validate the results by the original study. The reproduction is considered to be successful, if it demonstrates similar profit gains, price trajectories and punishment mechanisms as the original study.[2]

[2]In our implementation of the Q-learning update rule, i utilize the corrected formula for the parameter ν, as pointed out by the reviewers regarding footnote 20 in Calvano et al. (2020). The correction ensures the parameter grid matches the theoretical specifications intended by the original authors.

3 Results

To validate the presence of algorithmic collusion, I focus my reproduction on three specific metrics that collectively define algorithmic collusion. First, the average profit gain (Δ) is examined to establish that the algorithms can reach supra-competitive prices (see Figure 1 and Table 1). Second, i analyze the Impulse-Response-Functions to ensure the mechanism of the collusion (see Figure 2). Finally, i reproduce the parameter stability heatmap to ensure that the collusion persist across a range of learning and exploration parameters (see Figure 4).

The reproduction verifies the core conclusion of Calvano et al. (2020) that pricing algorithms are capable of attaining and maintaining supra-competitive prices without explicit collusion. The training outcomes demonstrate profit gains of approximately 80-90% (see Table 1) across sessions, with agents showing consistent ability to preserve prices above the Nash equilibrium. Figure 1 illustrates that these supra-competitive prices are reached in almost all periods, with only sporadic deviations where the Nash price is approached during the learning process.

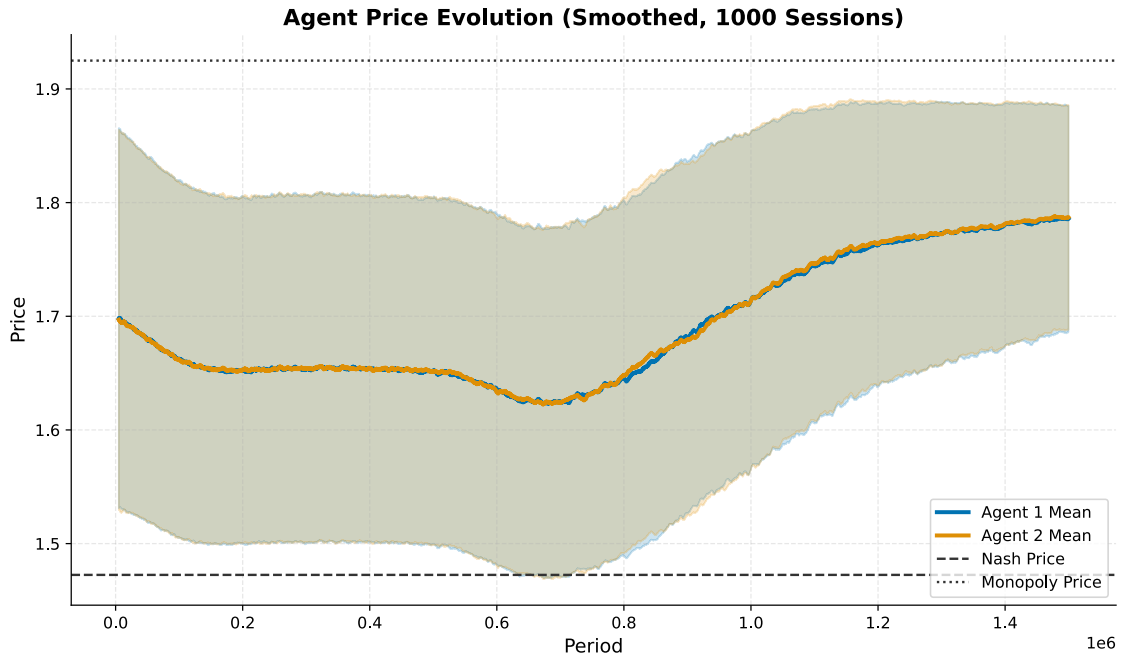

Figure 1: Price trajectories over time showing convergence to supra-competitive levels (reproduces Figure 10 in Calvano et al. (2020))

Notes: The algorithms maintain prices consistently above Nash equilibrium (1.47) and approach collusive levels (approximately 1.8) in most periods. The shaded area represents the one standard deviation confidence band (mean ± 1 std. dev.) across the 1,000 independent training sessions. The values are smoothed using a window size of 50.

The reproduction also gives support for the emergence of sophisticated punishment strategies.

An examination of the responses to a deviation in the two-agent case reveals that algorithms learn to enforce collusive behavior through targeted retaliation. As shown in Figure 2, when one agent deviates from the collusive price, the other responds with a price cut that undercuts the deviator. The agents then gradually return their prices to the long-run price, though with some remaining variance after period 9.

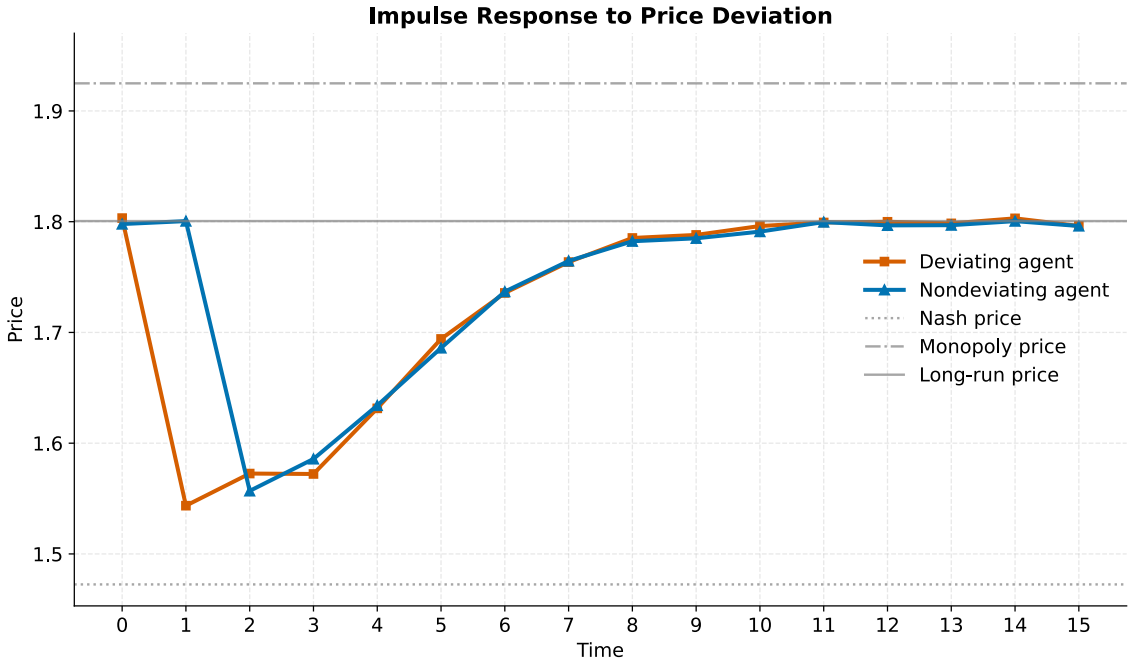

Figure 2: Punishment strategies following a deviation at period 1 (reproduces Figure 4 in Calvano et al. (2020))

Notes: After one algorithm deviates from collusive pricing, the competitor responds with a retaliatory price cut before both gradually return to supra-competitive levels.

The propensity for collusion depends heavily on the valuation of future profits. Figure 3 indicates that higher discount factors (δ) result in increased profit gains, reaching levels that exceed 0.6 when δ exceeds 0.85. The lowest profit gains occur at $\delta = 0.35$, with gains below 0.2, being in line with the findings by Calvano et al. (2020).

The impact of the learning parameters largely aligns with the original findings, though having some differences. While extremely low values of the exploration parameter β consistently yield lower profit gains (below 0.5), the effect of the learning rate α differs from the results by Calvano et al. The diagonal pattern of profitable parameter combinations remains, but optimal values occur at higher levels of α and β than in the original study (see Figure 4).

The propensity for algorithms to coordinate on supra-competitive prices is further validated by

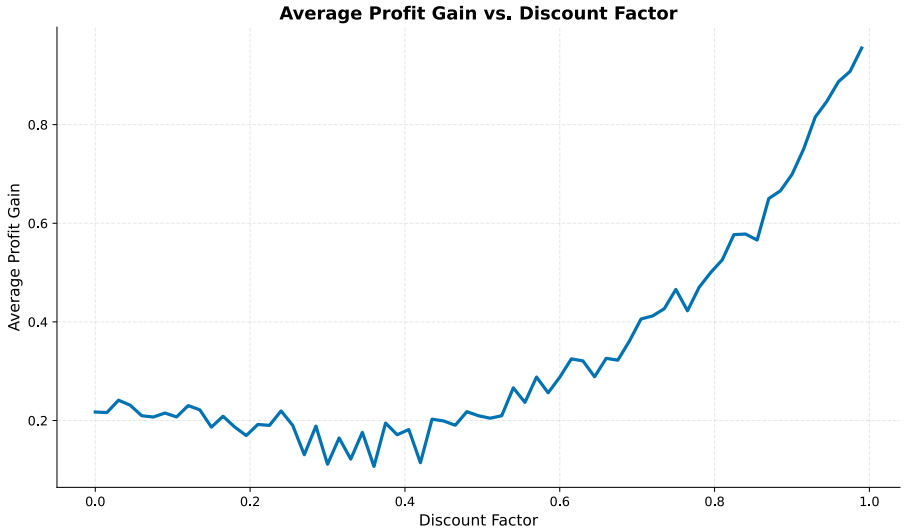

Figure 3: Impact of discount factor (δ) on profit gains from collusion (reproduces Figure 3 in Calvano et al. (2020))

Notes: Higher values of δ (greater weighting of future profits) lead to significantly increased profit gains, exceeding 0.6 when $\delta > 0.85$.

examining the frequency of convergence to the static Nash equilibrium. Figure 5 reproduces the convergence analysis of Calvano et al. (2020), plotting the fraction of sessions that converge to the competitive outcome across the (α, β) parameter space. I classify a session as converging to Nash if the final prices stabilize within a ± 2 grid-point tolerance of the theoretical Nash price.

The results in Figure 5 indicate that the collusive equilibrium is remarkably robust. While Calvano et al. (2020) identified a region of high Nash convergence at low learning rates, the results indicate that agents frequently learn to collude even in this region. This reinforces the conclusion that supra-competitive pricing is the dominant strategy for Q-learning algorithms in this environment, emerging robustly even under conditions theoretically favorable to competitive learning.

It is noteworthy that the convergence to stable strategies requires a substantial training period, having an average duration of 3.76 million periods (standard deviation: 139,004 periods). Final convergent prices range between 1.50 and 1.97 (mean: 1.802, median: 1.815, standard deviation: 0.086), suggesting robust supra-competitive pricing across training sessions.

In order to systematically evaluate the reproductions success, a comparison of key metrics with those by Calvano et al. (2020) is conducted. Table 1 presents this side-by-side comparison, summarizing the core quantitative outcomes.

As shown in Table 1, the core metrics align well. Both studies find profit gain of 70-90% above the competitive levels. The magnitude of collusive pricing is in line with the findings of the original

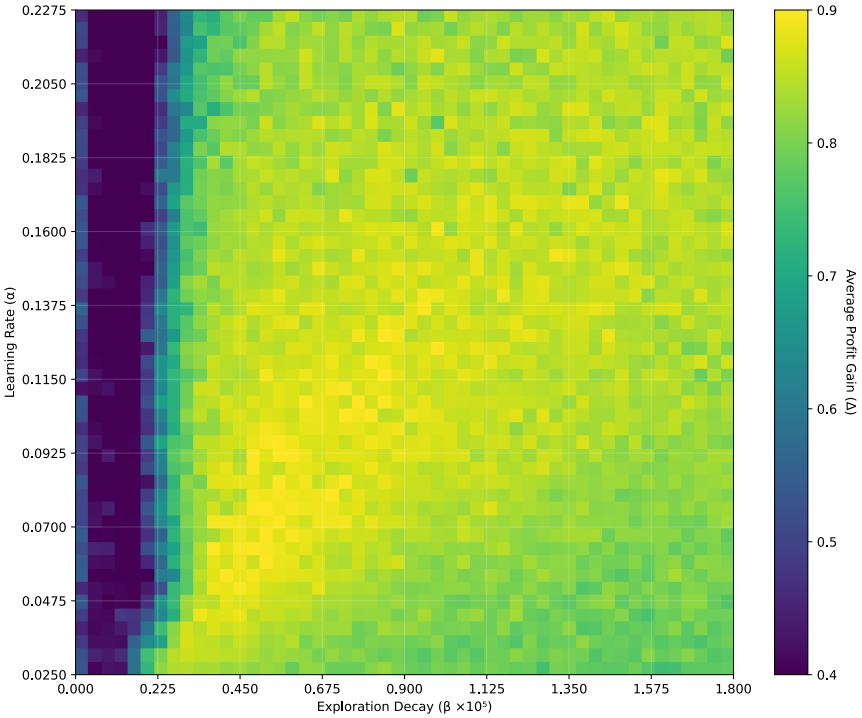

Figure 4: Relationship between learning parameters (α and β) and profit gains (reproduces Figure 1 in Calvano et al. (2020))

Notes: The diagonal pattern shows optimal combinations of learning rate (α) and exploration decay (β) for achieving collusive outcomes, with profit gains ranging from 0.8 to 0.9. A parameter grid of 50 values for α and β is used. The learning rate α (y-axis) was varied from 0.025 to 0.25, and the exploration decay β (x-axis) was varied from 0 to 2.0e-5. Each point in the grid uses the average of 100 sessions.

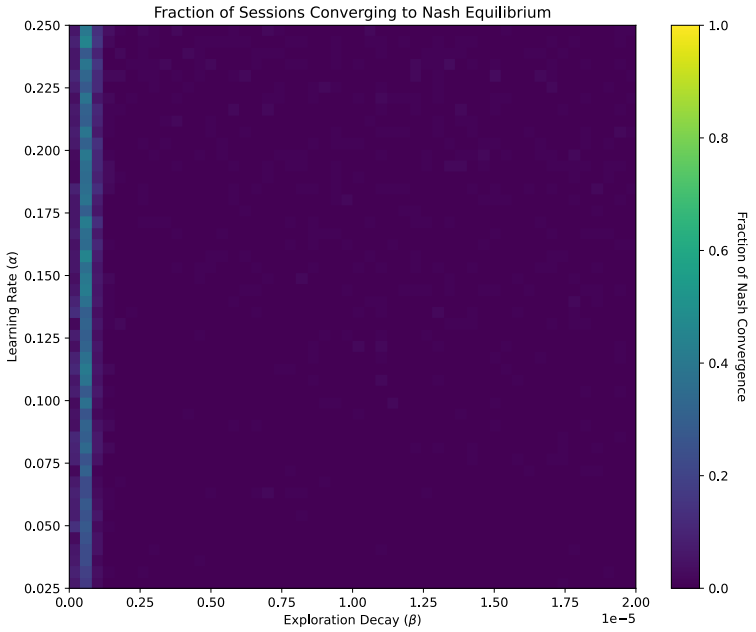

Figure 5: Fraction of sessions converging to the static Nash equilibrium (reproduces Figure 2 in Calvano et al. (2020))

Notes: Consistent with the original study, the competitive Nash outcome is an unstable attractor compared to the collusive equilibrium across most of the parameter space. Notably, my replication finds that collusion is even more robust than originally reported: while Calvano et al. observed high Nash convergence at low learning rates ($\alpha < 0.05$), our agents frequently locate collusive strategies even in this region, with Nash convergence peaking at only 45% in a narrow band of exploration parameters.

study. Both studies also exhibit similar price cuts after an external deviation by one of the agents.

Next to these similarities, there are some notable differences. The reproduction requires approximately 3.76 million periods to converge versus "*hundreds of thousands*" in the original study. While the findings for the exploration parameter β align well with the original study, some deviations can be found for the learning rate α, especially in the low values. Additionally, some differences are observable in the response after the punishment periods. These differences likely stem from minor implementation differences, as the core economic findings closely align.

Table 1: Side-by-side comparison of key findings

Metric	**Calvano et al. (2020)**	**Reproduction**
Equilibrium Profit Gain	70-90% above competitive levels. (Fig. 1, p. 3278)	Average Profit Gain: 86.3% (Std.Dev.: 12.3%), Median Profit Gain: 89.5%
Mean Convergent Price	Finding confirmed as "in line" with original study, but specific value not stated in text.	1.802 (std. dev. 0.086).
Periods to Convergence	From about 400,000 to several millions. (Online Appendix A3.1)	Approx. 3.76 million (std. dev. 139,004).
Effect of Discount Factor (δ) on Profit Gain	Confirmed to be consistent with original findings.	
- *At* $\delta = 0.35$	Profit gains below 0.2.	Profit gains at 0.11 at $\delta = 0.36$.
- *At* $\delta > 0.85$	Profit gains exceed 0.8.	Profit gains exceed 0.8 at $\delta = 0.92$.
	(Fig. 3, p. 3281)	(Figure 3 in the paper)
Punishment Mechanism	Finite phase of punishment followed by a gradual return to cooperation (Fig. 4, p. 3282).	Confirmed; deviator is undercut before a gradual return to the collusive price. (Figure 2 in the paper)
Optimal Learning Parameters (α, β)	Diagonal pattern of profitable combinations identified (Fig. 1, p. 3278).	Diagonal pattern confirmed, but optimal values occur at slightly higher levels of α and β. (Figure 4 in the paper)
Nash Convergence Frequency	High convergence at low α (Fig. 2).	Lower convergence observed; collusion persists even at low α (Figure 5 in the paper).

4 Robustness checks and extensions

Following Calvano et al.'s robustness check for asymmetric firms, this section also employs a robustness check to look at asymmetric firms. Asymmetries are introduced for costs of production and quality of production for the two different firms.

4.1 Robustness to asymmetric costs

The reproduction confirms the conclusion made by Calvano et al. (2020) that Q-learning algorithms are able to cope with the issue of asymmetric markets. But the outcomes for different prices of the second firm show a non-linear relationship. Based on the analysis of $N = 2000$ runs, two distinct phases can be identified:

- First phase: **Symmetric decline** (Cost 1.00 - 1.375 for the second firm): For small differences, the profit gains for both firms fall symmetrically. This means that the collusion is a little more difficult, but the equilibrium remains stable and symmetric.
- Second phase: **Efficient firm's advantages** (Cost 1.5 - 2.0 for the second firm): In the second phase, the first firm, which is more efficient, starts to exploit its cost advantage. The profit gain raises significantly from 0.755 to 0.791. Conversely, the profit gain of the second firm decreases to 0.694, though it recovers to 0.752 at $cost = 2.0$. This suggests that the algorithms learn a new strategy, where the low-cost firm leverages its competitive advantage to get a larger share of the profits, though the higher cost firm manages to regain some profit at higher cost imbalances.

4.2 Robustness to asymmetric qualities

This section reproduces the robustness analysis by Calvano et al. (2020) for the demand asymmetry, which they present in their Appendix. To test for demand asymmetry, the product quality index a_i is varied.

The results, presented in Table 2, reveal a significant discrepancy when compared to the original study. While the profit gains in the case of Calvano et al. (2020) remain below 1, this replication finds that the profit gain (Δ) for the low-quality firm (firm 1 in this case) increases disproportionately as the asymmetry becomes larger, exceeding 1.0 at quality 2.2 and reaching $\Delta = 9.414$ when Firm 2's quality is 2.4 (driven by the JPM denominator approaching zero).

This supports the result of a methodological difference that highlights a limitation of the Δ metric itself. The original Fortran code reads pre-calculated Nash and Coop-Prices from an external file to serve as benchmarks. The Python implementation in this project calculates the theoretical benchmarks (Nash and Joint Profit Maximum (JPM)) for every asymmetric setup using a numerical optimizer.

Our method and the results indicate that the JPM becomes a unstable coordination point when the asymmetry becomes large. In the JPM, the low-quality firm, in this case firm 1, would have to accept profits near or below the Nash equilibrium to maximize total aggregate profit. This causes

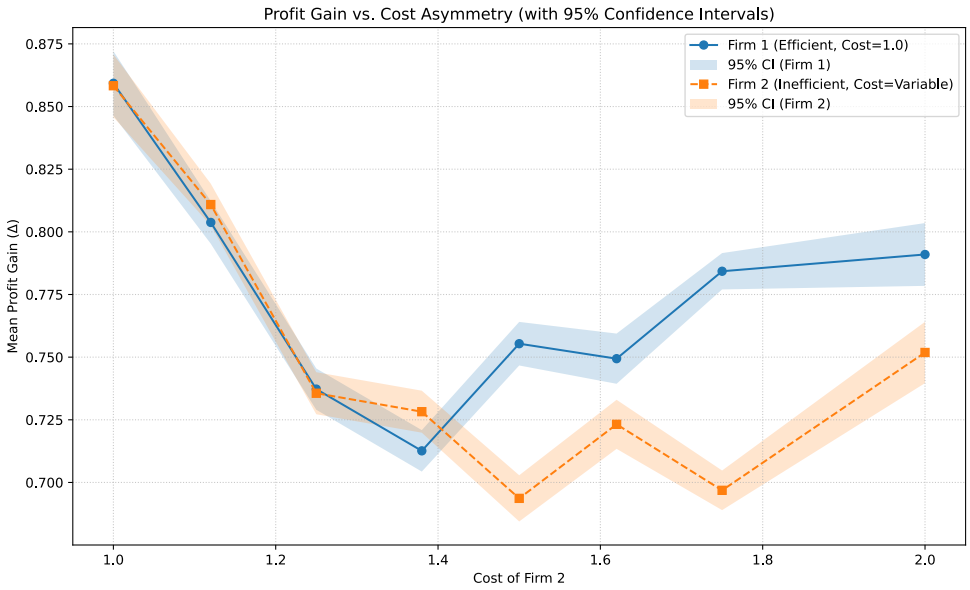

Figure 6: Mean profit gain vs. cost asymmetry

Notes: The plot shows the mean Δ for the efficient (Firm 1) and inefficient (Firm 2) firms, with shaded areas representing the 95% confidence intervals based on 2000 sessions. Detailed numbers can be found in Table 5.

the denominator of the profit gain calculation to approach zero for firm 1 leading to potentially high values for the profit gain.

The finding, which matches with the paper's finding, is that the algorithms do not converge to the Nash equilibrium. This is demonstrated by firm 2, which, in the scenario of $a_2 = 2.4$, still achieves a profit gain of 0.467. So these firms coordinate on a different, stable collusive outcome, which is still preferable to competition.

Furthermore, the rate of convergence increases with the asymmetry rising from 26.7% at quality level 1.6 to the highest level of 83.2% at quality level 2.4 . This suggests that the set of stable collusive equilibria is reduced by making the JPM a not-reachable target. This then simplifies the coordination problem, making it easier and faster for the agents to find a stable, collusive outcome.

Table 2: Results for different quality levels of firm 2

Quality Firm 2	Overall Avg. Δ	Firm 1 (Δ_1)	Firm 2 (Δ_2)	Convergence Rate (%)
1.6	0.772	0.921	0.622	26.7%
1.8	1.020	0.826	1.213	34.6%
2.0	0.860	0.860	0.861	63.4%
2.2	1.098	1.613	0.583	82.8%
2.4	4.940	9.414	0.467	83.2%

Notes: Firm one remains on the quality level of 2.0. The firms can maintain a high level of profit gain ($\Delta > 0.7$ for all price levels). Results are based on $N = 1000$ sessions for each quality configuration keeping all parameters for the agents the same.

5 Discussion

This study is able to reproduce the key finding by Calvano et al. (2020) that pricing algorithms learn to collude without explicit programming or communication. The reproduction confirms that these algorithms are able to achieve supra-competitive prices, with profit gains matching the gains in the original study at 70-90% above competitive levels.

The reproduction validates all major conclusions of the study by Calvano et al. (2020). The algorithms converge to similar price levels (mean: 1.802), exhibit comparable profit gains and sophisticated mechanisms of punishment. The price trajectories following deviations follow a similar pattern of immediate retaliation before a gradual return to collusive pricing, thereby confirming the robustness of the original findings.

Multiple notable differences emerge between this reproduction and the original study that warrant further discussion. Firstly, the convergence of the price trajectories in this implementation requires approximately 3.7 million periods, whereas the original study reported "*hundreds of thousands*" as sufficient to reach convergence. While this does not alter the conclusion about collusion, the difference is likely attributable to specific implementation choices detailed in Table 3. Despite the use of numpy and multiprocessing, the wall-clock runtime in Python is expected to be slower than in the original Fortran code, as Python is an interpreted language and Fortran is a compiled language. Furthermore, the use of numpy.random instead of the original's custom ran2 algorithm means the sequences of random exploration are different, potentially leading agents down longer learning paths before a stable equilibrium is found. Because the convergence criterion needs 100,000 *consecutive* periods of stable strategies, this metric is highly sensitive to the random number generating process. This represents a standard trade-off for Python's high-level accessibility and integration with the broader data science ecosystem. I chose Python over R or Julia due to its extensive adoption in both economics and machine learning, which provides a rich ecosystem of libraries (e.g., PyTorch, JAX) for the advanced extensions (like Deep Q Networks (DQNs)) that this paper helps facilitate. Secondly, post-punishment price trajectories exhibit greater variance after the return to the collusive price level compared to the trajectories by Calvano et al. (2020).

The impact of the discount factor δ on collusive behavior is consistent with the original findings. Both studies show that the profit gains are below 0.2 at δ-levels of 0.35 and raise up to 0.8, as δ exceeds 0.92. This relationship between weighting of future profits and collusive outcomes underlines the theoretical mechanism of algorithmic collusion.

Another significant difference appears in the relationship between the learning parameters (α and β) and the profit gains. The diagonal pattern of profitable parameter combinations identified by Calvano et al. was verified, but for extreme values of α, the profit gains in this reproduction (0.75-0.825) are higher than reported in the original study (below 0.7). The heatmap (Figure 4) shows a large region with profit gains lower than 0.4, which sharply transitions to profit gains of 0.8-0.9 in the diagonal. A deeper investigation of the final converged outcomes of 100 sessions at six points in the alpha-beta-spectrum is conducted. The results show a 'convergence cliff' instead of a phase change.

As shown in Appendix Table 4, the agents fail to converge for low values of beta after 5,000,000

periods per session, with convergence rates of 6% for $\beta = 1e-6$, respectively. The low profit gain ($\Delta \leq 0.4$) is a direct result of this failure because the agent's exploration phase is too long and thereby never come to a stable policy. At higher levels of β (i.e. $2.5e-6$), the convergence rate jumps to about 90%, allowing high profitable equilibria to be found by the agents. For the other representative points ($\beta = 1.5e-5$ or $\beta = 1e-5$), the convergence rates are between 87% and 90% but each experiment converged to a wide array of final outcomes (between 43 and 48 unique price pairs). This confirms the highly stochastical nature of the final collusive outcome. The implication of these findings is that stable collusion is sensitive to parameter tuning and the decay rate β is a very important parameter for finding a converged, high-profit equilibrium.

Furthermore, my replication results concerning the frequency of Nash convergence bear significant implications for antitrust policy. While Calvano et al. (2020) identified a region of the parameter space, specifically at low learning rates, where algorithms were more likely to compete than collude, my results suggest that this "competitive region" is narrower than originally reported. I find that algorithmic collusion is a remarkably robust attractor, emerging frequently even under conditions theoretically favorable to competitive convergence. This resilience implies that simple regulatory remedies, such as limiting the learning speed of pricing algorithms, may be insufficient to prevent tacit collusion, as the algorithms demonstrate a persistent ability to coordinate on supra-competitive prices across a broader range of the hyperparameter space.

The significant training period required for convergence in this study warrants a discussion of the practical time scale of learning, a point raised by Calvano et al. (2020). While an average of 3.76 million periods might seem impractical for real-time market adaptation, this long learning horizon does not diminish the policy concern. As it is noted by the authors, firms can apply extensive offline training in simulated environments before deploying an already-collusive algorithm into the live market. Furthermore, Calvano et al. (2020) note that supra-competitive prices are already achieved before the stable equilibrium is reached. The inherent slowness of Q-learning, which updates only one cell per period, necessitates these long training periods. Future research is already beginning to use more efficient algorithms, such as Deep Q Networks (DQNs) or Recurrent Neural Networks (RNNs), as seen in recent works by Hettich (2021) and Deng et al. (2025) for DQNs.

This successful reproduction further supports the conclusion by Calvano et al. (2020) regarding the concern of algorithmic collusion in markets. The finding of consistent maintenance of supra-competitive prices by basic tabular Q-learning agents could have an impact on competition policy and shows that existing regulatory frameworks are not equipped to handle these new challenges. As these algorithms are being increasingly used in markets like Amazon, competition authorities may need to consider "new forms of antitrust intervention" (Calvano et al., 2020). This could involve empowering agencies to "subpoena and test pricing algorithms" to search for collusive code, or exploring proactive regulatory solutions such as mandatory "transparency requirements, auditing of algorithms or design constraints" (Calvano et al., 2020) to ensure pricing algorithms are designed to foster, rather than inhibit, competition.

6 Conclusion

This study validates the findings by Calvano et al. (2020) on algorithmic collusion by a successful reproduction in a new environment. The results confirm the core finding that algorithms are capable to learn to collude without explicit instruction or communication, reaching supra-competitive prices and profit gains of 70-90%. Furthermore, the algorithms develop sophisticated punishment mechanisms to enforce the collusion, responding to external deviations with a price cut before gradually returning to collusive prices. Minor differences can be found in the speed of convergence and in the optimal parameter combinations for the learning parameters α and β. This is likely to be explained by implementation differences as the central conclusion is not undermined.

The translation of the Fortran implementation to Python helps support the ongoing expansion for research on algorithmic competition, answering the call by Calvano et al. (2020) for "more efficient" and "advanced" algorithms. By providing a validated baseline in an accessible language, this paper's framework can facilitate the active line of research that is already examining more advanced algorithms, such as DQNs (Hettich, 2021) and other deep reinforcement learning approaches (Deng et al., 2025). These advanced models can provide additional insights how algorithmic pricing could affect market outcomes and may address the "time scale of collusion" by learning more rapidly. Additionally, these models could be able to cope with more complex market environments.

The findings highlight the need for competition authorities to carefully assess how existing regulatory frameworks can address the challenges posed by algorithmic collusion. As firms increasingly delegate pricing decisions to algorithms, the confirmed potential of these algorithms to tacitly collude raises question about future competition policy. Future research in this area could explore specific interventions like transparency requirements, feasibility of auditing of algorithms or design constraints for pricing algorithms to maintain competitive price levels in the era of algorithmic pricing.

Bibliography

Calvano, E., Calzolari, G., Denicolò, V. & Pastorello, S. (**2020**). "Artificial Intelligence, Algorithmic Pricing, and Collusion". *American Economic Review*, 110(10): 3267–3297. doi:10.1257/aer.20190623. ISSN 0002-8282. URL https://www.aeaweb.org/articles?id=10.1257/aer.20190623.

Courthoud, M. (**2021**). "Replication of Artificial Intelligence, Algorithmic Pricing, and Collusion". URL https://github.com/matteocourthoud/Algorithmic-Collusion-Replication.

Deng, S., Schiffer, M. & Bichler, M. (**2025**). "Exploring Competitive and Collusive Behaviors in Algorithmic Pricing with Deep Reinforcement Learning". *arXiv preprint arXiv:250311270*. URL https://arxiv.org/abs/2503.11270.

Epivent, A. & Lambin, X. (**2024**). "On algorithmic collusion and Reward–punishment Schemes". *Economics Letters*, 237: 111661. DOI: 10.1016/j.econlet.2024.111661.

Eschenbaum, N., Mellgren, F. & Zahn, P. (**2022**). "Robust Algorithmic Collusion". *arXiv preprint arXiv:220100345*. URL https://arxiv.org/pdf/2201.00345.

Hettich, M. (**2021**). "Algorithmic Collusion: Insights from Deep Learning". *Available at SSRN 3785966*. URL https://www.wiwi.uni-muenster.de/cqe/sites/cqe/files/CQE_Paper/cqe_wp_94_2021.pdf.

Lewis, J. (**2024**). "AlgorithmicCompetition.jl". URL https://github.com/jeremiahpslewis/AlgorithmicCompetition.jl.

Werner, T. F. (**2021**). "qpricesim: Quantum Pricing Simulation Tool". URL https://github.com/ToFeWe/qpricesim.

yusei406 (**2024**). "Calvano Strict Replication: Replication Code for the Calvano Thesis". URL https://github.com/Yusei406/calvano-thesis. MIT Licensed.

A Appendix

Table 3: Implementation comparison between original and reproduction

Component	**Calvano et al. (Fortran)**	**Python Reproduction**
Core Architecture	Tabular Q-Learning with parallel processing using OMP	Tabular Q-Learning with numpy arrays
State Space	Fixed length states with integer arrays	Numpy arrays representing states
Action Selection	Two mechanisms supported: (1) Epsilon-greedy (2) Boltzmann exploration	Epsilon-greedy only
Q-Matrix Updates	Updates single cell at a time with critical sections for parallel access	Vectorized updates using numpy operations
Convergence Check	Checks if strategy remains constant for 100,000 consecutive periods	Same approach but with numpy array comparisons
Memory Management	Explicit array allocation and deallocation with garbage collection calls	Automatic memory management via Python/numpy
Parallelization	OpenMP directives for parallel processing across sessions	Process-based parallelization using Python multiprocessing
Random Number Generation	Custom implementation of ran2 algorithm	numpy.random
Data Storage	Direct file I/O with formatted writes	Pandas DataFrames for data manipulation and storage
Performance Optimization	Manual loop optimizations, critical sections for parallel access, explicit memory management	Vectorized operations, numpy's optimized C backend, automatic memory management

Table 4: Configuration and results for convergence analysis

α	β	**Periods**	**Convergence Rate**	**Unique Final States**
0.150	1.0e-6	5,000,000	6% (6/100)	4
0.150	1.5e-6	5,000,000	91% (91/100)	26
0.150	2.5e-6	5,000,000	100% (100/100)	34
0.150	1.0e-5	5,000,000	90% (90/100)	48
0.225	1.0e-5	5,000,000	87% (87/100)	48
0.150	1.5e-5	5,000,000	87% (87/100)	43

Notes: All experiments were run with $\delta = 0.95$ and 100 sessions. The "Unique Final States" column counts the number of distinct price pairs found among the sessions that successfully converged.

Table 5: Results for different cost levels of Firm 2

Cost Firm 2	Firm 1 (Δ_1)	95% CI (Δ_1)	Firm 2 (Δ_2)	95% CI (Δ_2)
1.0	0.859	0.013	0.858	0.013
1.125	0.804	0.009	0.811	0.008
1.25	0.737	0.008	0.736	0.008
1.375	0.713	0.008	0.728	0.009
1.5	0.755	0.009	0.694	0.009
1.625	0.755	0.010	0.723	0.010
1.75	0.784	0.007	0.697	0.008
2.0	0.791	0.013	0.752	0.012

Notes: Firm one remains on the cost level of 1.0. The profit gains are calculated using asymmetric benchmarks. The confidence intervals are 95% confidence intervals. Results are based on $N = 2000$ sessions for each cost configuration keeping all parameters for the agents the same.