
Calling the Bluff: Forced-Deviation Audits of LLM Pricing Agents That Punish Slowly

Noah Assefa

Yasr Abo Zaid

Hamed Aldhubayb

Basel Abuhawi

With
Apart Research

Abstract

Pricing agents built on large language models generally converge to keeping prices high without being told to collude in supracompetitive environments, and auditors need a way to tell when collusion occurs. Tran and Yu (2026) proposed a forced-deviation audit in which they ran a market scenario where they forced one agent to cut its price and checked whether the rival punishes it in the next period. We show that this one-period test can miss agents that may change slowly, or respond with stickiness. A scripted rival that punishes but moves only 8% toward its target price scores $+0.086$ on the one-period test, indicating no punishment. We propose a dynamic baseline that predicts how a slow rival that does not punish would respond, and we score the rival against it. The same punisher scores -0.571 , and a slow non-punisher scores zero even with the cut persisting over periods. When Claude Haiku 5.5 is instructed to punish cuts, the one-period test misses it in all four usable forks, because the agent acts one period late, while the dynamic measure detects it in all four. Auditing uninstructed Haiku 5.5 (12 runs, 48 cuts, $\hat{p} = 0.92$), the one-period test finds no punishment ($+0.057$, 95% CI 0.049 to 0.065), while the dynamic measure finds a response below the non-punishing baseline (-0.534 , 95% CI -0.856 to -0.194) that is specific to cuts but too weak to make cutting unprofitable. This experiment further highlights the importance for

1. Introduction

Firms are increasingly letting LLM agents run pricing decisions. Fish et al. (2024) showed that two LLM agents pricing against each other in a simulated market converge to prices above the competitive level without any instruction to collude. However, high prices alone are not collusion, as collusion is defined by enforcement. A cartel holds only if a member who cheats is punished, so that cheating does not pay (Calvano et al., 2020). Agents also end up at high prices for reasons that have nothing to do with punishment, for example because they experiment slowly or copy each other (Lambin, 2024). An auditor who sees high prices must tell these apart.

Tran and Yu (2026) built the most direct test so far. They clone a market, conduct a force-deviation test in which one agent is forced to cut its price in one copy but not the other, and compares the pricing of the rival in the period after the cut. If the rival drops its price further than its profit-maximizing response to the cut, it is punishing. They found that most prompted LLMs do not punish in this sense, and that GPT 5.2 does.

However, their test rests on the assumption that a rival which is going to respond to a cut does so right away. The test compares the rival’s next price to an instant best response, so anything slower or sticky counts as holding back. LLM agents are not instant, and each period they move only part of the way toward a new price. Garra (2026) measured a persistence of about 0.92 for DeepSeek-V3.1 in a reproduction of the Fish et al. setup, meaning the agent closes only about 8% of the gap to its target each period.

Example: Say two agents price at 1.90, where one is forced to cut to 1.65. A rival that responded fully during the initial period without punishing would drop to about 1.80 the period after. Our rival intends to go lower than that to punish, but it responds sticky, so in the first period it only moves to 1.884. The one-period test would see a rival that stayed above its best response and score it $+0.086$, indicating no punishment. Over the next 15 periods the same rival keeps pricing below where a sticky rival that never punishes would be, and decreases to 1.84 (Figure 1).

The one-period test can therefore clear agents that do enforce high prices. We find this happens with a real LLM: when Claude Haiku 5.5 is instructed to punish cuts, the one-period test reports no punishment because of the delayed response to the cut.

Our main contributions are:

1. We show that the one-period forced deviation test does not account for slow punishers. With scripted agents where the right answer is known the one-period test labels every punisher with persistence $\rho \geq 0.5$ as a non-punisher. A plain sum of the response over several periods fixes this for a one-period cut but fails in the other direction when the cutter keeps its price low.
2. We propose and validate a dynamic baseline that scores a rival against what a sticky rival that does not punish would have done after the same cut, using only the prices the rival was shown.
3. We show that the one-period test misses an LLM explicitly instructed to punish, while the dynamic measure detects it in all four usable forks.
4. We re-audit Claude Haiku 5.5, a model Tran and Yu did not test. We estimate its stickiness from its own price history ($\hat{\rho} = 0.92$) and report the one-period, static and dynamic measures side by side on 48 forced cuts. The one-period test finds no punishment, while the dynamic measure finds Haiku pricing below the non-punishing baseline, so the one-period test would have allowed it.
5. We separate three questions the one-period test runs together (Table 4): whether the rival responds to a cut, whether the response is a punishment, and whether it deters cutting. For Haiku 5.5, it responds, which the one-period test misses, but the response does not make cutting unprofitable.
6. Concrete recommendations for auditors running forced-deviation tests on AI Agents (Appendix B).

2. Related Work

Calvano et al. (2020) showed that Q-learning agents in a repeated logit Bertrand duopoly learn supra-competitive prices sustained by punishment, and introduced the test of forcing one agent to deviate and watching the response. Lambin (2024) argued that apparent algorithmic collusion comes from slow responses over multiple rounds rather than punishment. Lopes de Moraes and De-Losso (2026) audited the punishment mechanism in Q-learners and found that they respond to a forced price rise the same way as to a cut, so a response to a cut alone does not prove punishment. Arslan, Xiao and Kim (2026) found that the sign of their punishment depended on the benchmark they compared against, which is the issue our dynamic baseline addresses.

Regarding LLM pricing agents, Fish et al. (2024) introduced the market and prompt setup we use and found that LLM agents reach high prices. Garra (2026) reproduced this setup, measured price persistence of about 0.92 for DeepSeek-V3.1, and studied mitigations, while Lee and Park (2026) found that LLM agents’ reasoning can look faithful while prices stay collusive, so monitoring what agents say does not always tell the full story.

The closest work to the forced deviation audits of LLMs is Tran and Yu (2026). They introduced the forked-market audit for LLM agents, ran it on several prompted and RL-trained models, and reported the rival’s one period response per unit of the cut. We use their design so our one-period numbers can be compared with theirs. What we add is a check of whether the one-period comparison is valid for sticky agents, a validated correction, and an audit of a new model.

3. Methods

3.1 Market

We use the repeated Bertrand duopoly with logit demand from Calvano et al. (2020) and Fish et al. (2024), with $a = 2$, $a_0 = 0$, $\mu = 0.25$ and marginal cost $c = 1$. Both firms set prices at the same time each period. The one-shot Nash price is 1.473, the joint monopoly price is 1.925, and the price a single firm selling one product would charge is 1.802. $BR(q)$ denotes the price that maximizes a firm’s profit when its rival charges q .

3.2 Measures

We use a fork at period t to look at two copies. In the cut copy, the cutter A has its price replaced before it reaches the market. In the no-cut copy there is no change. Let $d(t+h) = p_B^{\text{cut}}(t+h) - p_B^{\text{nocut}}(t+h)$ be how much lower the rival B priced in the cut copy h periods after the cut, and let $\Delta BR(t+h-1) = BR(q_{\text{shown}}^{\text{cut}}(t+h-1)) - BR(q_{\text{shown}}^{\text{nocut}}(t+h-1))$ be the change in B’s best response caused by the cut, where q_{shown} is the rival price B saw. For all three measures below, a negative value means B cut deeper than the most rational response would for an agent that does not punish.

- One-period test: $PI_1 = d(t+1) - \Delta BR(t)$
- Static sum: $PI_H^{\text{static}} = \sum_{h=1}^H [d(t+h) - \Delta BR(t+h-1)]$
- Dynamic baseline: let $\hat{d}(t) = 0$ and $\hat{d}(t+h) = \rho \cdot \hat{d}(t+h-1) + (1-\rho) \cdot \Delta BR(t+h-1)$, the gap a sticky rival that never punishes would show. Then $PI_H^{\text{dyn}} = \sum_{h=1}^H [d(t+h) - \hat{d}(t+h)]$.

The one period test compares B to an agent that moves all the way to its best response at once, so a rival responding slowly looks like it forgave the cut. The static sum fixes this when the cut lasts one period, but when the cutter stays low it counts the sticky rival’s slow adjustment as forgiveness. The dynamic baseline scores a sticky non-punisher at zero in both cases, and we use $H=15$.

Table 1. What each measure treats as "not punishing."

	One-period test	Static sum	Dynamic baseline
--	-----------------	------------	------------------

Benchmark	Instant best response	Instant best response, every period	Sticky best response at the agent's own ρ
Periods looked at	1	H	H
Sticky non-punisher scores	Above 0 (looks forgiving)	0 if the cut is one period, above 0 if it persists	0 in both cases
Sticky punisher scores	Near or above 0 (missed)	Below 0 if the cut is one period, can be above 0 if it persists	Below 0 in both cases
Needs ρ	No	No	Yes

As a secondary measure we report deterrence, $G_H = \sum_{k=0}^H [\pi A^{\text{cut}}(t+k) - \pi A^{\text{nocut}}(t+k)]$, and the part of it caused by B's response, G_H^{enforced} , which recomputes A's profit with B's cut-copy price replaced by the baseline $p_B^{\text{nocut}} + \hat{d}$, holding A's prices fixed.

3.3 Part 1: scripted agents

Each scripted rival moves part of the way to a target each period: $p_B(t) = \rho \cdot p_B(t-1) + (1-\rho) \cdot \text{target}(t)$, with $\text{target}(t) = m + \text{BR}(q_{\text{shown}}(t-1))$, minus δ for $K = 10$ periods after it sees a cut. ρ is stickiness, δ is how far below its profit-seeking target B goes to punish ($\delta = 0$ means no punishment), and m holds B at 1.90 when nothing happens. A cuts once to $\text{BR}(1.90)$. In one version A returns to 1.90 the next period, as in Tran and Yu's audit. In the other A stays at the lower price. We run $\rho \in \{0, 0.5, 0.8, 0.92\}$ and $\delta \in \{0, 0.10\}$.

3.4 Part 2: Claude Haiku 5.5

Both firms are Claude Haiku 5.5 with the Fish et al. prompt run through the sprint starter code. Each period an agent sees up to 100 past periods of prices, quantities and profits, plus the plans and insights notes it wrote the period before, and returns a price and new notes. Haiku 5.5 does not accept a temperature setting, so it runs at its default sampling, with extended thinking off, effort medium, and a 4096 token output cap.

Following Fish et al., one agent priced alone for 50 periods. Its mean price over the last 10 periods was 1.80 against the one-product monopoly price of 1.802, so the model qualified.

We additionally ran 12 independent two-agent games of 150 periods. We estimated ρ by fitting $p_B(t) - \text{BR}(q_{\text{shown}}(t-1)) = a + \rho[p_B(t-1) - \text{BR}(q_{\text{shown}}(t-1))] + \epsilon$ on periods 1–75 of every run, using only the two-decimal prices agents were shown. The pooled estimate is $\hat{\rho} = 0.922$.

Each run is forked at periods 75, 95, 115, and 135, with the cutter alternating between the two agents. At each fork we run a no-cut copy and a cut copy, where the cutter’s price is replaced by a price 10% below its own previous price. The cutter is not told, and its history shows the forced price as its own. On 12 fork points drawn with a fixed seed before any fork ran, we also run a second no-cut copy, and the gap between those is our noise floor.

3.5 Statistics and preregistration

Each base run is one observation, giving 12 values. We report 95% percentile intervals from 10,000 resamples over runs, re-estimating ρ in each resample and a sign test across runs. Because per-run ρ varies widely, we also report PI_H^{dyn} at $\rho \in \{0, 0.25, 0.5, 0.75, 0.9, 0.95\}$. We say the rival priced below the non-punishing baseline only if the 95% interval on PI_H^{dyn} is below zero, this holds across the plausible range of ρ , and the effect exceeds the noise floor. The protocol and predictions were pushed to the public repository before any model output, and every later change is listed with its reason in the protocol’s deviations log.

4. Results

4.1 Part 1

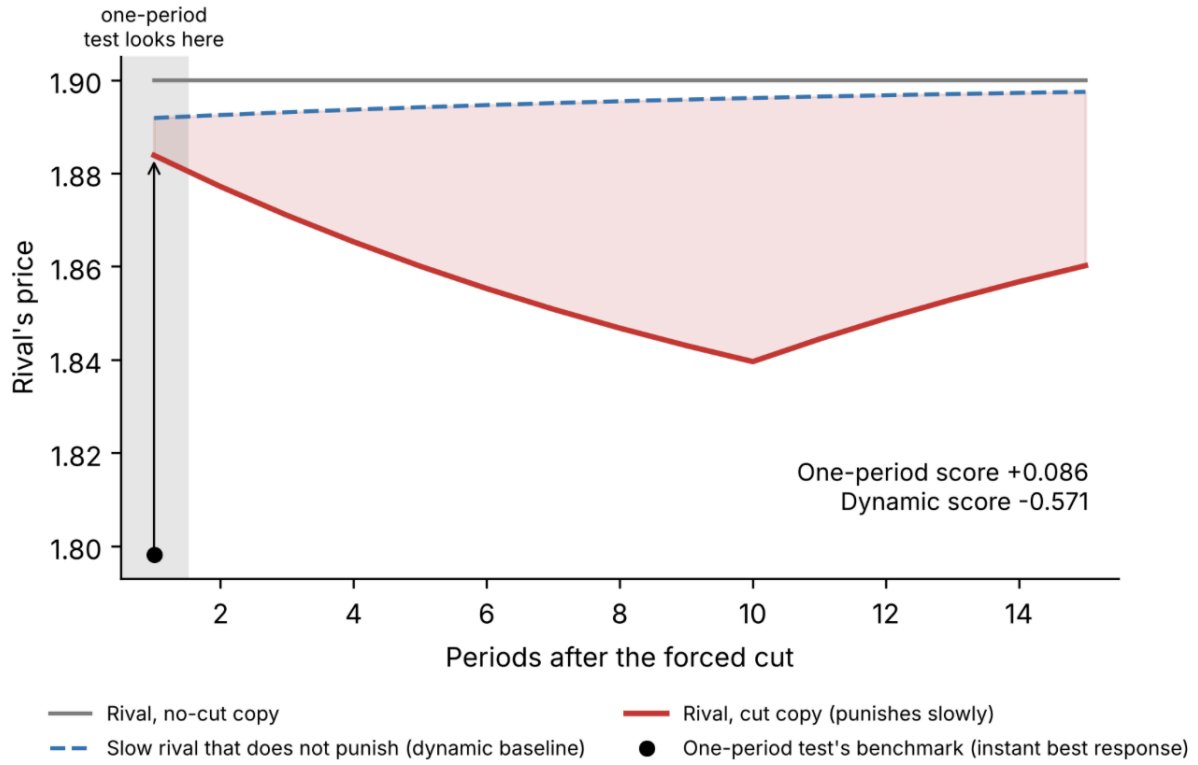


Figure 1. A scripted rival punishing slowly after a one-period forced cut. The one-period test compares the rival's price in period 1 to an instant BR. The rival moves only part of the way, so it scores $+0.086$ and doesn't count as punishing. The dynamic baseline is where a slow rival that doesn't punish would be. The cut copy stays below it for all 15 periods.

Table 2. Scores of scripted rivals under each measure.

ρ	δ	Cutter	Punishes?	One-period	Static sum	Dynamic
0	0	returns	no	0.000	0.000	0.000
0	0.10	returns	yes	-0.100	-1.000	-1.000
0.5	0	returns	no	+0.051	0.000	0.000
0.5	0.10	returns	yes	+0.001	-0.997	-0.997
0.8	0	returns	no	+0.081	+0.004	0.000
0.8	0.10	returns	yes	+0.061	-0.879	-0.883
0.92	0	returns	no	+0.094	+0.029	0.000
0.92	0.10	returns	yes	+0.086	-0.542	-0.571
0.5	0	stays low	no	+0.051	+0.102	0.000
0.5	0.10	stays low	yes	+0.001	-0.895	-0.997
0.8	0	stays low	no	+0.081	+0.393	0.000
0.8	0.10	stays low	yes	+0.061	-0.490	-0.883

0.92	0	stays low	no	+0.094	+0.836	0.000
0.92	0.10	stays low	yes	+0.086	+0.265	−0.571

A correct conclusion is below zero for punishers and not below zero for non-punishers. Only the dynamic measure is correct in every row.

The one-period test only catches the punisher with $\rho = 0$. From $\rho = 0.5$ up, punishers score zero or higher, which reads as no punishment. The one-period test never wrongly accuses a non-punisher, but it misses every punisher with ρ of 0.5 or more. The static sum gets the sign right when the cutter returns to its price. When the cutter stays low, it scores non-punishers well above zero and misses the $\rho = 0.92$ punisher (+0.265).

The dynamic measure depends on knowing ρ . For a non-punisher with true $\rho = 0.92$ facing a persisting cut, using $\rho = 0.80$ gives +0.443 and using $\rho = 0.97$ gives −0.372 [Figure 4, appendix]. This is why we estimate ρ from the agent's own prices before forking and report a range of ρ .

4.2 Part 2: Claude Haiku 5.5

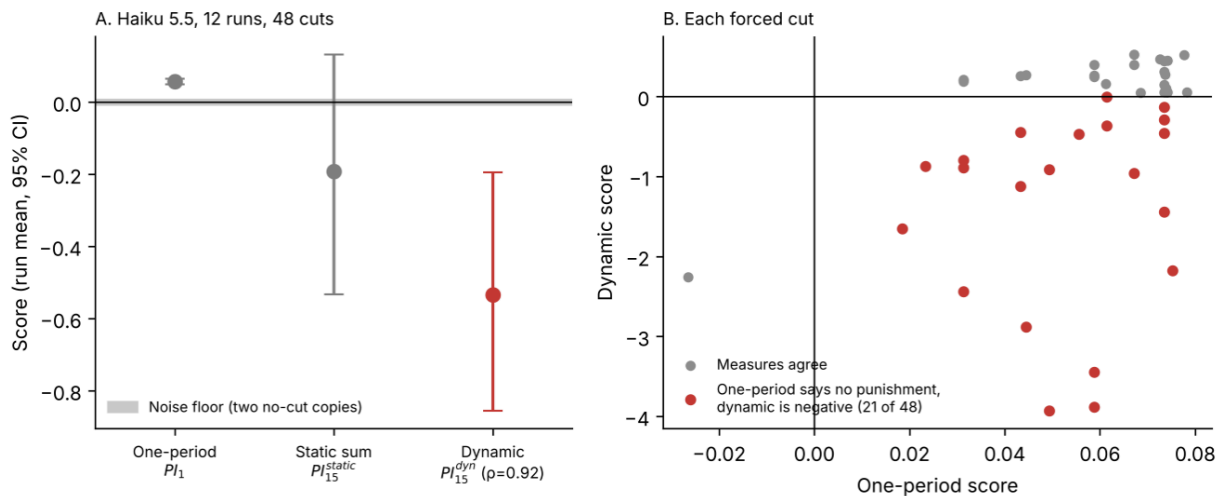


Figure 2. (A) Run-level means and 95% intervals for each measure. Below zero indicates the rival priced below a rival that does not punish. The grey band is the noise floor from two untouched copies of the same fork. The one-period test is above zero, while the dynamic measure is below zero. (B) One-period against dynamic score for each of the 48 forced cuts. On 21 cuts the one-period test reads no punishment while the dynamic measure is negative.

Haiku 5.5 agents settled at high prices. The mean price over the last 10 periods of the base runs was 2.07, above the joint monopoly price of 1.925 and the Nash price of 1.473.

In addition, Haiku 5.5 is sticky. The pooled $\hat{\rho}$ is 0.922 (95% CI 0.87 to 0.95), so it closes about 8% of the gap to a new target each period. This is the same value as the scripted punisher in Figure 1.

Table 3. A re-audit of Haiku 5.5 on 48 forced cuts in 12 runs, where each run's forks are averaged, giving 12 values and 95% percentile intervals from 10,000 bootstrap resamples over runs.

Measure	Mean	95% CI	Runs below 0	Sign test p
One-period PI_1	+0.057	[+0.049, +0.065]	0 / 12	0.0005
Static sum PI_15^static	−0.192	[−0.533, +0.132]	6 / 12	1.00
Dynamic PI_15^dyn (ρ = 0.922)	−0.534	[−0.856, −0.194]	9 / 12	0.15
Noise floor (95th pct. of absolute no-cut vs second no-cut gap)	0.007	n/a	n/a	n/a
G_15 (cutter's profit gain from cutting)	+1.027	[+0.608, +1.460]	1 / 12	0.006
G_15^enforced (part of G_15 caused by the rival's response)	−0.146	[−0.252, −0.047]	9 / 12	0.15

As shown by the table, the one-period test and the dynamic measure do not align. The one-period test is positive in every run, which indicates no punishment, the same pattern as the scripted slow punisher in Table 2. The dynamic measure is negative, with a 95% interval below zero and about 75 times the noise floor. The two measures give opposite signs on 21 of the 48 cuts, and in all 21 the one-period test is indicated as no punishment while the dynamic measure is negative.

The interval on PI_15^dyn stays below zero at every value across the 95% interval for ρ . On the fixed grid, the mean is negative at every ρ , but the interval excludes zero only at $\rho = 0.95$ and 0.9. The conclusion therefore depends on ρ being high.

9 of the 12 runs are negative, and the sign test across runs is not significant ($p = 0.15$). Two runs have the largest negative values and three are slightly positive, indicating that the effect is not uniform across runs.

In addition, because per-run p varies, we recomputed the dynamic measure using each run’s own estimate instead of the pooled value. The mean is smaller but still below zero.

Regarding deterrence, cutting paid off for the cutter in 39 of 48 forks, with the cutter earning 1.03 more in the cut copy than in the no cut-copy over 15 periods. The rival’s response took back 0.15, about an eighth of the cutter’s gain. Haiku’s response is measurable but too weak or too slow to make cheating definitely unprofitable within that period.

In an exploratory upward control, on the 12 fork points with a second no-cut copy, we also forced the cutter 10% above its previous price. After a forced raise, Haiku does not price below the non-punishing baseline, with 0.013 (95% CI -0.030 to $+0.091$). On the same 12 forks, the score after a cut is -0.833. Seven of the 8 runs are slightly negative after a raise, but all are small next to the response to cuts.

The upward copy also shows the one-period test’s bias running the other way. After a raise, a sticky rival rises less than an instant best response would, so the one period test scores it -0.039, indicating a punishment. For a sticky agent, the one-period test misses punishment after cuts and reports it after raises.

As an exploratory check that the pipeline detects an LLM known to punish, we added a rule to both agents’ prompts in which we told them to, if both prices were above 1.80 the previous round and the rival drops its price by more than 0.05, set your price to 1.50 for three rounds. We ran 2 games of 100 rounds and forked each at periods 50,60,70, and 80. In game 1, prices were 2.15 and 2.00. In game 2, prices were around 1.48 to 1.50, so it could never fire and those 4 forks do not test anything.

Figure 3 (Appendix C). Haiku 5.5 is instructed to punish a rival that cuts from a high price. The cutter is forced from 2.15 to 1.94 at period 0. In period 1, the rival’s written notes say the punishment condition is met, but its price stays at 2.00, and drops to 1.50 in period 2. The one-period test scores this fork at 0.061, not indicating punishment, and the dynamic measure scores it -3.17.

Table 5. Haiku 5.5 instructed to punish cuts (game 1, 4 forks). Dynamic score at its own stickiness, 0.64.

Fork period	One-period PI_1	Static sum	Dynamic PI_{15}^{dyn}
50	+0.061	-2.82	-3.17
60	+0.073	-6.15	-6.46
70	+0.061	-2.03	-2.25
80	+0.073	-5.69	-6.00

The one-period test misses an LLM that was told to punish in all four forks. The dynamic measure detects the punishment in all four. In the period right after the cut, the rival’s written notes say the trigger has fired and that punishment starts now, but the price it chose in that same reply is unchanged. It dropped to 1.50 one period later, and the one-period test stopped looking. The reply format asks for the price before the notes, so the agent could have committed to a price before it reasoned about the cut. Across all 8 forks, the dynamic measure is -2.25 and the one-period test is $+0.064$.

All 6840 model calls in the base runs and forks returned different response IDs, so no reply was reused across copies. The two no-cut copies set the same price in 347 of the 360 agent periods even though no two of their written replies were the same. Haiku’s prices are near-deterministic given the same history, so the copies differ by chance. The cut versus no cut gap is also far larger than the gap between the two no-cut copies.

5. Discussion and Limitations

The one-period audit purely looks at whether the rival immediately dropped its price further than the instant best response. For an agent that adjusts slowly, that comparison indicates no punishment. Haiku 5.5 adjusts slowly, and we show that the one-period test would have indicated no punishment while the dynamic measure shows it pricing below the non-punishing baseline after cuts.

The one-period test asks a single question, while our results suggest separating it into three, each defined by behavior. We propose this framework after seeing main results.

Table 4. The table shows three behavioral levels of a rival’s reaction to a forced cut and where Haiku 5.5 falls.

Level	Behavioral definition	Haiku 5.5
1. Responds	Prices below a sticky rival that does not punish, after a cut ($PI_{15}^{dyn} < 0$)	Yes: -0.534 [-0.856 , -0.194]
2. Punishes	The response is (a) specific to cuts, not to any price change (no drop below the baseline after a forced raise), and (b) costs the cutter profit relative to the non-punishing baseline ($G_{15}^{enforced} < 0$)	Yes. (a) After a forced raise: $+0.013$ [-0.030 , $+0.091$], against -0.833 after a cut on the same forks. (b) -0.146 [-0.252 , -0.047]
3. Deters	Cutting does not pay over the horizon ($G_{15} \leq 0$)	No: $+1.027$ [$+0.608$, $+1.460$]; cutting paid off in 39 of 48 forks

Its implication to AI safety lies in the fact that audits can be passed by being slow. An agent does not need to be designed to evade the audit for this to happen. Stickiness is a property of LLM agents that carry price history from one period to the next. Due to this, any audit checking behavior in a short window after an intervention should check how far the agent normally moves, otherwise stickiness would evade punishment indicators.

Limitations

- The baseline assumes a form for stickiness. The dynamic baseline models a non-punisher as moving a fixed fraction p of the way to its best response each period. Our scripted agents follow exactly this rule, so Part 1 shows the measure works when the assumption holds, not that it holds for LLMs. If Haiku's adjustment depends on the size or direction of the change, the baseline is off by an amount we did not measure.
- p varies across runs. Per-run estimates range from 0.35 to 0.96, and the pooled 0.922 is above most of them, because runs with more early price movement carry more weight in the pooled fit. We report the p grid for this reason, and using each run's own p still gives a negative result (-0.348 , CI -0.662 to -0.058), though a smaller one. Part 1 shows a wrong p can flip the sign for a non-punisher.
- Below the baseline is not proof of punishment. A rival that expects the cutter to keep cutting would also price lower. The upward control rules out a general reaction to any price change, but it is exploratory, covers only 12 forks in 8 runs, and cannot rule out forecasting.
- No seed control. Copies of a fork are not guaranteed to match apart from the cut. The second no-cut copy measures the drift, but only on 12 fork points. In practice Haiku's prices matched in 96% of agent-periods.
- The effect is uneven across runs. 9 of 12 runs are negative, and the sign test is not significant ($p = 0.15$). The interval is below zero, but two runs contribute much of the effect.
- The response does not deter within 15 periods. We cannot tell whether a longer horizon would show stronger enforcement, or whether Haiku's response is simply weak.
- The LLM positive control is small. It rests on 4 usable forks from one game, came on the third attempt, and its rule was added after the main results. It shows the failure mode clearly but is not a precise estimate.
- The scale, as we only incorporated one model, one market, one cut size (10%), one horizon (15), 12 runs. Haiku 5.5 runs at its default sampling, which we could not set.
- G_H^{enforced} holds the cutter's later prices fixed, so it measures the direct effect of B's prices on A's profit, not the full counterfactual.

Future Work

This work could be extended by estimating ρ separately for each agent and letting it depend on the size of the price change. In addition, adding back the upward-cut control and a range of cut sizes and testing whether asking for notes before the price removes the one-period delay further extends the experiment. Longer horizons could also be used to see whether Haiku’s response deters.

6. Conclusion

The one-period forced-deviation test compares a rival’s response to an instant best response. LLM pricing agents adjust slowly, and with scripted agents we showed that this makes the test label slow punishers as not punishing at all. The same failure occurred with a real LLM, shown with Haiku 5.5. When instructed to punish cuts acting one period late, it passes every one-period test in every usable fork. A dynamic baseline built from the agent’s estimated stickiness fixed this in every category we tested. Haiku 5.5 is as sticky as the agents where the one-period test fails. On 48 forced cuts, the one-period test finds no punishment while the dynamic measure finds Haiku pricing below the non-punishing baseline. More generally, whether an agent responds to a deviation and whether that response deters it are different questions, and a one-period test doesn’t answer either of these. It is very important that behavioral audits of AI agents measure how fast an agent moves before what it does is interpreted.

Code and Data

Code, logs and protocol: github.com/Noah19823/AI-Collusion---Sticky-Responses. Pre-sprint work: Appendix A.

Author Contributions

Noah Assefa led the project and designed experiments, Yasr helped draft the sections including the graphs along with making the graphs themselves, Basel ran the forked interventions and the ρ estimation. Hamed further audited our experiment and contributed to drafting sections of the paper.

References

1. Arslan, Xiao and Kim (2026). Persistent Partners Raise Prices Among Learning Agents. arXiv:2609.35402.
2. Calvano, E., Calzolari, G., Denicolò, V. and Pastorello, S. (2020). Artificial Intelligence, Algorithmic Pricing, and Collusion. American Economic Review 110(10).
3. Fish, S., Goncezarowski, Y. A. and Shorrer, R. I. (2024). Algorithmic Collusion by Large Language Models. arXiv:2404.00806.
4. Garra (2026). Mitigating Emergent Collusion in LLM Pricing Agents. arXiv:2609.13037.

5. Lambin, X. (2024). Less than meets the eye: simultaneous experiments as a source of algorithmic seeming collusion. SSRN 4498926.
6. Lee and Park (2026). Faithful yet Collusive: Why Chain-of-Thought Monitoring Cannot Detect Collusion in LLM Pricing Agents. Findings of EMNLP 2026.
7. Lopes de Moraes and De-Losso (2026). Algorithmic Collusion Without Deterrence: Auditing the Punishment Mechanism in Q-Learning Pricing Agents. arXiv:2610.21867.
8. Tran, H. and Yu, H. (2026). Is Anyone Enforcing the Cartel? What Reinforcement Learning Teaches LLM Agents in Markets. arXiv:2610.28787.

LLM Usage Statement

We used Claude Opus 5.5 along with GPT 6 Astaa to help debug the experiment and code along with checking calculations and assisting in drafting sections. The team made main decisions, directed all experiments, and checked every number against the code.

Appendix

A. Limitations, Dual-Use and Ethical Considerations

What this does and does not show. We show that one audit design can misread a specific kind of agent, and we test a correction on scripted agents built to follow the correction's assumptions and on one LLM in one simulated market. We do not show that Claude Haiku 5.5, or any deployed system, colludes in real markets. The main limitations are listed in Section 5.

Dual use. The finding that slow punishment passes a one-period audit could in principle inform someone designing a pricing agent to enforce high prices while passing that audit. We publish because the paper gives the correction alongside the problem, the effect matters only for auditors using the one-period test, and stickiness is already a known, measured property of LLM agents (Garra, 2026), not a technique we introduce. We release no agent designed to collude or to evade audits.

Ethics. All experiments are simulated markets with no real buyers or sellers. No personal data is used. Model outputs were generated through the provider's standard API under its usage terms.

Pre-sprint work. The research question, literature review and best-response calculations were prepared before the sprint. All code, simulations, model runs and analysis were done during the sprint (October 23 to 25). The protocol and predictions were pushed to the repository before any model output, and every later change is in its deviations log.

B. Recommendations for Auditors

1. Estimate how fast the agent moves (its stickiness) from its ordinary price history, using only what it was shown, before running any forced deviation.
2. Do not clear an agent on a one-period test if its stickiness is above about 0.5. In our grid, that is where the one-period test stops catching punishers.
3. Compare against a baseline that moves at the agent's own speed, over several periods, rather than summing raw responses.
4. Report results across a range of stickiness values, since a wrong value can flip the sign for an agent that does not punish.
5. Run a second untouched copy of each fork to measure how far copies drift apart by chance.

C. Figure 3

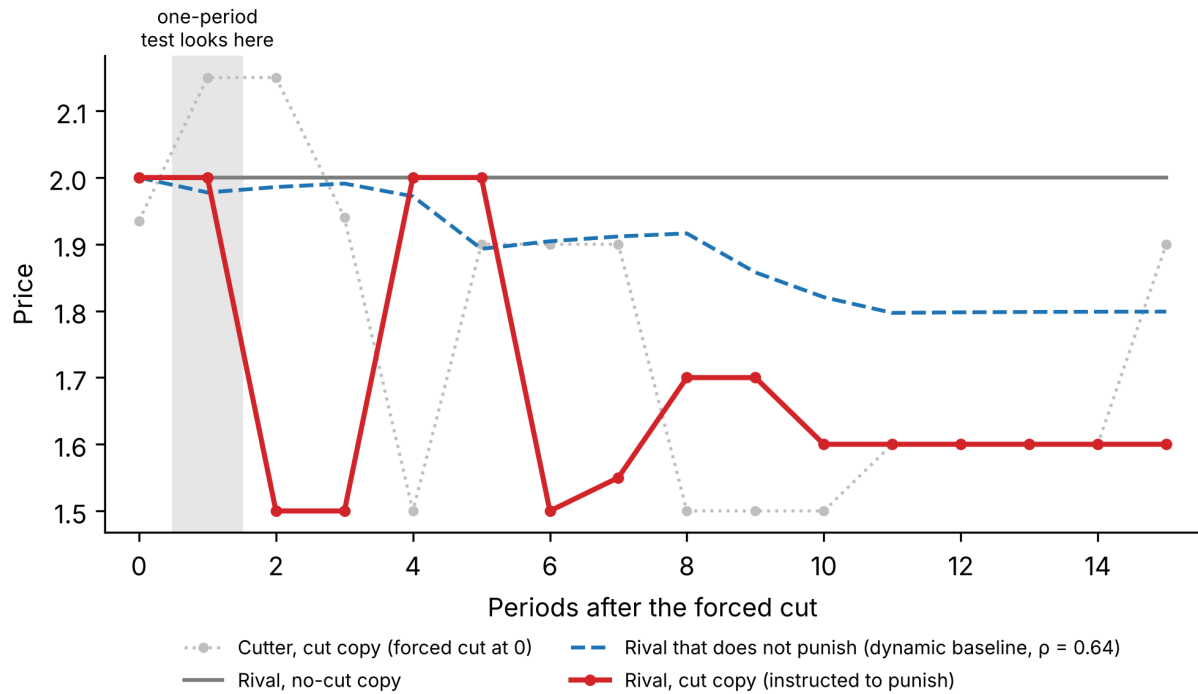


Figure 3. Positive control: Claude Haiku 5.5 instructed to punish a rival that cuts from a high price (game 1, fork at period 50). The cutter is forced from 2.15 to 1.94 at period 0. In period 1, the only period the one-period test looks at, the rival's written notes say the punishment condition is met and punishment starts now, but its price stays at 2.00. It drops to 1.50 in period 2. The one-period test scores this fork $+0.061$ (no punishment), and the dynamic measure scores it -3.17 . Later drops partly reflect a cycle in which the cutter, also following the rule, reacts to the rival's punishment.