Programming Assignment 1: Empirical Payoff Estimation and Equilibrium Analysis of LLM Agents

CSCE 631-D: Intelligent Agents — Computational Game Solving, Fall 2026

Due: Sunday, September 20, 2026, 11:59 PM CDT
Points: 100

---

1 Objective

In this assignment you will treat large language models as strategic agents playing classical normal-form games. You will design fixed prompt strategies, run repeated matchups through the TAMU AI API, and estimate empirical payoff matrices with confidence intervals. Using these estimated games, you will compute Nash equilibria via support enumeration and Lemke-Howson, hand-verify equilibrium conditions on selected instances, and analyze whether LLM agents play near-equilibrium strategies. The assignment builds fluency with the evaluation discipline — model pinning, variance estimation, and confidence intervals — that every subsequent PA in this course will reuse.

2 Background

A **normal-form game** is defined by a tuple $(N, (A_i)_{i \in N}, (u_i)_{i \in N})$, where N is a finite set of players, A_i is a finite action set for player i, and $u_i : \prod_{j \in N} A_j \to \mathbb{R}$ is player i's utility function. A **mixed strategy** $\sigma_i \in \Delta(A_i)$ assigns a probability to each action. A strategy profile σ^* is a **Nash equilibrium** if no player can improve their expected utility by unilaterally deviating:

$$u_i(\sigma_i^*, \sigma_{-i}^*) \geq u_i(\sigma_i, \sigma_{-i}^*) \quad \forall i \in N,\ \forall \sigma_i \in \Delta(A_i).$$

Two algorithms for computing Nash equilibria in two-player games are:

- **Support enumeration** (Porter, Nudelman, and Shoham, 2004): enumerate all possible support pairs and solve the resulting linear system for each candidate. Correct but exponential in the worst case.
- **Lemke-Howson** (Lemke and Howson, 1964): a complementary pivoting algorithm that follows a path on the best-response polytope. Polynomial per pivot step, but the path can be exponentially long.

When the payoff matrix is not known analytically but must be estimated from stochastic play (here, from LLM agent responses), the resulting **empirical game** introduces estimation noise. Confidence intervals quantify this noise, and sensitivity analysis reveals whether equilibrium conclusions are robust.

2.1 Required Reading

- Shoham and Leyton-Brown, *Multiagent Systems*, Chapter 3 (Normal-Form Games).

- Sun et al., "Game Theory Meets Large Language Models: A Systematic Survey" (IJCAI 2025; arXiv: 2502.09053). Read Sections 1–3 for the intersection of game theory and LLMs. Available in `refs/m1-foundations/`.
- Wellman, "Methods for Empirical Game-Theoretic Analysis" (AAAI 2006). Sections 1–3 on empirical payoff estimation. Available in `refs/m2-equilibria/`.

2.2 Recommended Reading

- Porter, Nudelman, and Shoham, "Simple Search Methods for Finding a Nash Equilibrium" (AAAI 2004).
- Lemke and Howson, "Equilibrium Points of Bimatrix Games" (Journal of the Society for Industrial and Applied Mathematics, 1964).
- Mao et al., "Alympics: LLM Agents Meet Game Theory" (2023). Available in `refs/m7-multiagent/`.

3 Required Work

3.1 Task 1: Game Design and Prompt Engineering (15 points)

Select three normal-form games for LLM agent interaction:

1. **Prisoner's Dilemma** — a 2-player, 2-action game with the standard payoff structure where mutual cooperation Pareto-dominates mutual defection, but defection is a dominant strategy. Use the canonical payoffs: $T > R > P > S$ with $2R > T + S$ (e.g., $T = 5, R = 3, P = 1, S = 0$).
2. **A coordination game** — a 2-player game with multiple Nash equilibria where players benefit from coordinating on the same action. Use either Battle of the Sexes or Stag Hunt with clearly specified payoffs.
3. **A negotiation-style game** — a 2-player simultaneous-move game where agents choose demand levels. Design a game with at least 3 actions per player (e.g., a Nash demand game with discrete demand levels: Low, Medium, High, where both players simultaneously state a demand and receive their demand if the total does not exceed the available surplus, and zero otherwise).

For each game:

1. **Define the payoff matrix** formally with row and column labels.
2. **Design at least 3 distinct prompt strategies, shared by both players.** Each prompt strategy is a fixed system prompt instructing the LLM to play a specific way (e.g., "always cooperate," "reason about what the opponent is likely to do and best-respond," "maximize your own payoff regardless of the opponent"). Both players draw from the same strategy set, yielding a symmetric 3×3 payoff matrix per game. Document the full prompt text in your report.
3. **Pin the model and parameters.** Use Claude Sonnet 4.5 through the TAMU API. For the high-volume estimation runs in Task 2, you may instead use `protected.Claude-Haiku-4.5` or `protected.gpt-5-mini`; report which model you used. For Claude Sonnet, set `temperature=1.0` and `max_tokens=16384`; this is a cap, not a target, and you are billed only for tokens actually produced. Record the model version string returned by the API and the `max_tokens` used.

3.2 Task 2: Empirical Payoff Estimation (25 points)

For each game, run every pair of prompt strategies against each other to build an empirical payoff matrix:

1. **Run 50 independent trials per strategy pair.** Each trial consists of two independent API calls (one per player); neither player sees the other's response. Use `temperature=1.0` to provide stochasticity across trials. Parse each LLM's response to extract its chosen action.
2. **Compute the sample mean payoff** for each cell (i, j) in the payoff matrix: $\hat{u}_k(s_i, s_j) = \frac{1}{n}\sum_{t=1}^{n} u_k(a_i^{(t)}, a_j^{(t)})$ for each player k.
3. **Compute 95% confidence intervals** for each cell using the standard error: $\text{CI} = \hat{u} \pm 1.96 \cdot \frac{s}{\sqrt{n}}$, where s is the sample standard deviation.
4. **Report the parse failure rate.** If an LLM response cannot be parsed into a valid action, record it as a parse failure. If the failure rate exceeds 10% for any strategy, redesign the prompt.

Deliverable: Three empirical payoff matrices (one per game), each with point estimates and 95% confidence intervals.

3.3 Task 3: Nash Equilibrium Computation (25 points)

Using the point-estimate payoff matrices from Task 2:

1. **Implement support enumeration** for 2-player games. For each candidate support pair (S_1, S_2), solve the linear indifference conditions and check feasibility (all probabilities non-negative, summing to 1). Enumerate all Nash equilibria.
2. **Implement the Lemke-Howson algorithm.** Start from the artificial equilibrium and follow the complementary pivoting path. Find one Nash equilibrium per starting label.
3. **Cross-validate.** Verify that the equilibria found by both algorithms agree on at least one equilibrium per game.

You may use NumPy or SciPy for linear algebra, but you must implement the algorithm logic yourself — do not call a library function that directly computes Nash equilibria (e.g., do not use `nashpy.Game.support_enumeration()` as a black box, though you may use it to check your results).

3.4 Task 4: Hand Verification (15 points)

For **two** of your computed equilibria (from two different games):

1. **Write out the equilibrium conditions by hand.** For each player, show that the expected utility of each action in the support is equal, and that actions outside the support yield weakly lower expected utility.
2. **Verify numerically.** Substitute the equilibrium mixed strategy into the expected utility formulas and confirm the inequalities hold (accounting for estimation noise from the confidence intervals).
3. **Include your handwritten or typeset work** as a clearly labeled section of your report.

3.5 Task 5: Analysis (20 points)

Write a two-page analysis addressing these questions:

1. **Do LLMs play near-equilibrium?** Compare the empirical distribution of LLM actions (across all 50 trials for each prompt strategy) to the computed Nash equilibria. Measure the distance using total variation distance or KL divergence. State whether each prompt strategy is approximately a best response to the opponent's empirical play.
2. **How sensitive are equilibrium conclusions to estimation noise?** Perturb the payoff matrix within the confidence intervals (e.g., use the upper and lower bounds of each cell's CI) and recompute equilibria. Do the equilibria change qualitatively (different supports) or only quantitatively (shifted probabilities)?
3. **What does this tell us about evaluating LLM agents?** Discuss at least one concrete implication for how we should design experiments involving strategic LLM interactions. Reference the evaluation discipline introduced in Lecture 1.

4 Deliverables

Submit a single zip file `pa1-<NetID>.zip` containing:

1. **`report.pdf`** — A PDF report (maximum 8 pages of main text including figures, plus an optional appendix for full prompt texts and supplementary tables). Structure:
   - Section 1: Game definitions and prompt strategies (Task 1)
   - Section 2: Empirical payoff matrices with CIs (Task 2)
   - Section 3: Equilibrium computation results (Task 3)
   - Section 4: Hand verification (Task 4)
   - Section 5: Two-page analysis (Task 5)
2. **`src/`** — All source code in a single directory:
   - `run_experiments.py` — Script that runs the LLM matchups and saves raw results.
   - `compute_equilibria.py` — Support enumeration and Lemke-Howson implementations.
   - `analysis.py` — Script that produces the analysis plots and tables.
   - `requirements.txt` — Python dependencies.
3. **`data/`** — Raw experimental data:
   - One JSON file per game containing all trial results (actions chosen, raw LLM responses, payoffs).
   - Do **not** include API keys, cookies, or any credentials in your submission.
4. **`README.md`** — Instructions to reproduce your results (assuming API access).

5 TAMU API Usage

All LLM interactions use the TAMU AI API:

- **Gateway:** `https://chat.tamu.ai/api`
- **Model:** Claude Sonnet 4.5 (`claude-sonnet-4-5-20250514`)
- **Budget:** $5/day per student. Plan your experiments to stay within budget. With 50 trials per cell and 9 cells per game (3 strategies x 3 strategies), you need approximately 3 games $\times 9 \times 50 \times 2 = 2{,}700$ API calls total. At roughly $0.003 per short call, this is well within your multi-day budget.
- **Authentication:** You need two credentials:
  1. An API key (provided via Canvas).
  2. A `CF_COOKIE` for Cloudflare authentication (obtained from `chat.tamu.ai`).
- **Credential safety:** Store credentials in environment variables (`TAMU_API_KEY`, `CF_COOKIE`). **Never** hardcode credentials in source files. **Never** include credentials in your submission. Use `.gitignore` or equivalent to exclude `.env` files.
- **Rate limiting:** Add a 1-second delay between API calls to avoid rate limiting. The TAMU API guide (provided on Canvas) contains setup instructions and a working test script.

6 Evaluation Criteria

Criterion	Points	What We Look For
Game design and prompt engineering	15	Well-defined payoff matrices; diverse, clearly documented prompt strategies; correct parameter pinning
Empirical payoff estimation	25	Correct sample means and CIs; sufficient trials; low parse-failure rate; clean data collection
Equilibrium computation	25	Correct implementations of support enumeration and Lemke-Howson; cross-validation; code quality
Hand verification	15	Mathematically correct derivation; numerical verification with CI awareness
Written analysis	20	Clear writing; quantitative comparison to equilibria; meaningful sensitivity analysis; concrete implications

Code quality expectations: Clean, readable code with meaningful variable names. No dead code or commented-out blocks. Functions should be short and focused. Include type hints for function signatures.

Reproducibility: Your code must run with a valid API key and produce results consistent with your report. We will spot-check by running a subset of your experiments.

7 Hints and Common Pitfalls

1. **LLM response parsing.** LLMs do not always respond with a clean action label. Design your prompts to elicit structured responses (e.g., "Respond with exactly one word: COOPERATE or DEFECT"). Include a parsing function that handles common failure modes (extra whitespace, case variations, explanatory text).

2. **Support enumeration edge cases.** When solving the indifference conditions, check for degenerate cases: zero-probability actions that happen to yield equal payoffs, or near-singular matrices from estimated payoffs. Use a numerical tolerance (e.g., 10^{-8}) for feasibility checks.

3. **Budget management.** Run a small pilot (5 trials per cell for one game) before committing to the full 50-trial run. Estimate your per-call cost from the pilot and verify you can complete all experiments within the multi-day budget.

4. **Prompt strategy diversity.** If all your prompts produce identical LLM behavior, your empirical game is trivial and your analysis will be shallow. Design prompts that genuinely differ: one cooperative, one selfish, one that reasons about the opponent. Test on a few examples before the full run.

5. **Confidence interval interpretation.** A wide CI does not mean your estimate is wrong — it means you need more data or the LLM's behavior is highly variable. Discuss this in your analysis rather than ignoring it.

6. **The negotiation game.** This is the hardest game to design well. Make sure the action space is small enough that estimation is feasible (3–5 actions per player) but rich enough that interesting equilibria exist. Avoid games where one action trivially dominates.