Capstone Project Guide: "Markets and Machines"

1 Project Overview

This semester, we will simulate a "Wisdom of Crowds" experiment using AI agents. We will split into two teams to build competing market mechanisms: an **Automated Market Maker (LMSR)** and a **Continuous Double Auction (Order Book)**.

Our goal is to observe how agents with diverse, private beliefs and varying risk tolerances (CRRA) drive price convergence in these different market structures as new information becomes available.

2 The "Physics" of the World (Common to Both Teams)

Before splitting into teams, everyone must agree on these fundamental classes.

2.1 The Event (The "Ground Truth")

- **Type:** Binary Option.
- **Outcomes:** 1 (True/Heads) or 0 (False/Tails).
- **Payoff:** A share pays \$1.00 if the event occurs, \$0.00 otherwise.
- **Ground Truth (P^*):** The fixed, underlying reality (e.g., the coin has a bias of 0.65). This value is hidden from the agents but acts as the anchor for the information signals they receive.

2.2 The Agent (The "Trader")

Agents are algorithmic traders governed by **Constant Relative Risk Aversion (CRRA)** preferences.

1. Belief Initialization (p_i)

Each agent starts with an initial private belief p_i. To create market liquidity, we seed these beliefs from a distribution centered on the Ground Truth:

$$p_i \sim \text{ClippedGaussian}(\mu = P^*, \sigma = 0.1)$$

2. Trading Strategy

Agents do not use "gut feelings"; they trade to maximize Expected Utility. Based on the derivation in Sethi et al. (2024), the optimal trade size x^* (number of shares to buy or sell) is calculated as:

$$x^* = \frac{(k-1)y - z}{1 + q(k-1)} \tag{1}$$

Where:

- y = Current Cash
- z = Current Shares (can be negative/short)
- q = Current Market Price
- k is the risk-weighted "edge" defined as:

$$k \equiv \left(\frac{p(1-q)}{q(1-p)} \right)^{1/\rho} \tag{2}$$

- ρ (rho): The risk aversion parameter.
  - $\rho \approx 0$: High risk tolerance.
  - $\rho = 1$: Log Utility (Kelly Criterion).
  - $\rho > 1$: High risk aversion.

3 Python Starter Code: The Agent Class

Use this code to ensure both teams are using the exact same trading logic.

```python
import numpy as np

class CRRAAgent:
    def __init__(self, agent_id, initial_cash, belief_p, rho):
        self.id = agent_id
        self.cash = initial_cash        # y in the paper
        self.shares = 0                 # z in the paper
        self.belief = belief_p          # p in the paper
        self.rho = rho                  # Risk aversion parameter

    def get_optimal_trade(self, market_price):
        """
        Calculates optimal trade size x* based on Eq 8 in Sethi et al. (2024).
        Returns:
            x_star (float): Number of shares to buy (+) or sell (-).
        """
        q = market_price
        p = self.belief
        y = self.cash
        z = self.shares
        rho = self.rho

        # Avoid division by zero or log errors
        if q <= 0.01 or q >= 0.99:
            return 0.0

        # Case 1: Agent agrees with market (No trade)
        if abs(p - q) < 1e-6:
            # The paper notes that when p=q, the agent liquidates (x = -z).
            # However, for stability, we can simply hold (return 0).
            return 0.0

        # Calculate k (The risk-weighted edge)
        # k = ( (p * (1-q)) / (q * (1-p)) ) ^ (1/rho)
        numerator = p * (1 - q)
        denominator = q * (1 - p)
        k = (numerator / denominator) ** (1 / rho)

        # Calculate x* (Optimal Trade)
        # x* = ( (k-1)y - z ) / ( 1 + q(k-1) )
        x_star = ((k - 1) * y - z) / (1 + q * (k - 1))

        # --- SAFETY CHECKS (Bankruptcy Constraints) ---
        # Ensure the trade doesn't result in negative wealth in either outcome.

        # Max buy (limited by cash)
        max_buy = y / q if q > 0 else 0

        # Max sell (limited by "margin" - simplified)
        # Ensure we have liquidity to cover the short position
        if x_star > 0:
            x_star = min(x_star, max_buy)
        else:
            max_sell = y / (1-q) if (1-q) > 0 else 0
            x_star = max(x_star, -max_sell)

        return x_star

    def update_portfolio(self, trade_shares, trade_price):
        """
        Updates cash and shares after a trade is executed.
```

```python
        """
        cost = trade_shares * trade_price
        self.cash -= cost
        self.shares += trade_shares
```

4 Dashboard Interface + Deployability (New Requirement)

Each team must ship a lightweight dashboard that visualizes the live simulation and market state *while the simulation is running* (not just after-the-fact plots). The dashboard should connect to your market engine (LMSR "house" or CDA "exchange") via a small API layer and support repeatable runs from a fixed random seed. This turns the project into a product-style system rather than a standalone script and makes it easier to compare mechanisms under identical agent behavior and information regimes (Phases 1–3).

4.1 Minimum Dashboard Views (Both Teams)

At minimum, the dashboard must include:

1. **Market price over time**
   - A live chart of q_t each round.
   - Markers for major events (e.g. belief update rounds in Phase 2).
2. **Convergence / tracking view**
   - Display error vs. ground truth P^* (known to the simulator but not agents), e.g. $|q_t - P^*|$.
3. **Liquidity / microstructure (mechanism-specific)**
   - **LMSR:** show liquidity parameter b, outstanding share counts (or inventory vector), and instantaneous price impact (optional but encouraged).
   - **Order Book (CDA):** show best bid/ask, spread, last trade price, and a simple depth ladder (top N levels).
4. **Agent diagnostics**
   - A table of top agents by PnL and by traded volume.
   - Distribution plots (or summary stats) for cash, shares, and PnL, optionally sliced by risk aversion ρ.
5. **Run controls**
   - Start/stop a simulation run.
   - Choose scenario presets: Phase 1 / Phase 2
   - Set core parameters: number of agents, random seed, and (for Phase 2) signal noise level.

4.2 Recommended API Contract

To keep the dashboard decoupled from the engine, teams should expose a minimal HTTP API:

- `POST /runs` (create a run with config: seed, phase, `n_agents`, etc.)
- `POST /runs/{run_id}/start` and `/stop`
- `GET /runs/{run_id}/state` (current round, price, last trade, etc.)
- `GET /runs/{run_id}/metrics` (convergence error, volume, spread, etc.)
- `GET /runs/{run_id}/agents` (agent snapshots: cash, shares, belief, ρ, pnl)

This API contract must be compatible with both mechanisms, and encourages clean separation between market core logic and UI.

5 Team Assignments

5.1 Team A: Automated Market Maker (LMSR)

- **Mechanism:** Logarithmic Market Scoring Rule.
- **Role:** You are the "House." You define a cost function that allows agents to buy/sell instantly at a calculated price.
- **Key Math:**
  - Cost Function: $C = b\ln(e^{q_1/b} + e^{q_0/b})$
  - Price: $p_i = \frac{e^{q_i/b}}{\sum e^{q_j/b}}$

5.2 Team B: Order Book (Double Auction)

- **Mechanism:** Continuous Double Auction.
- **Role:** You are the "Exchange." You match buyers with sellers.
- **Challenge:** The agents calculate a specific quantity x^* to trade, but in an order book, they must also propose a **price**.
  - *Hint:* Since these agents calculate quantity based on the *current* market price, you can have them submit "Market Orders" (taking liquidity) or "Limit Orders" priced slightly favorable to their belief.

6 Roadmap

1. **Phase 1 (Static Beliefs):** Run the simulation with a fixed Ground Truth (e.g., $P^* = 0.70$) and static agent beliefs.
   - *Goal:* Do the agents trade until the market price converges to the mean of their initial beliefs?
   - *Analysis:* How does changing ρ (risk aversion) affect the final position sizes? (See Proposition 1 in the paper).
2. **Phase 2 (Updating Beliefs):** The Ground Truth P^* remains fixed, but the agents' *knowledge* of it improves over time.
   - *New Info:* In each round t, agents receive a noisy signal S_t (e.g., a "poll" or sample data point derived from P^*).
   - *Update:* Agents update their internal belief p_i using a weighted average or Bayesian update.
   - *Goal:* Observe how the market price tracks the "discovery" of the Ground Truth as agents trade on new information.
3. **Phase 3 (Analysis):** Using the **Profitability Test** described in the paper, evaluate which agents (high risk vs. low risk) perform better in each market type.

Reference

Sethi, R., Seager, J., Morstatter, F., Benjamin, D. M., Hammell, A., Liu, T., ... & Subramanian, R. (2024). *Political Prediction and the Wisdom of Crowds.* CI 25: Proceedings of the ACM Collective Intelligence Conference.