

# Discovering Equilibrium-like Behavior with LLM Agents: A Payment Systems Case Study

Hugi Aegisberg

December 22, 2025

## DO NOT CIRCULATE

This is a working document and is not intended for distribution.

This paper and the accompanying SimCash codebase were developed with assistance from **Claude 4.5 Opus** (Anthropic) and **GPT-5.2** (OpenAI) as coding and writing tools. All experimental design, analysis, and interpretation are the author’s own.

## About This Document

This is a **research proposal** presenting methodology and preliminary findings to potential collaborators. All tables, figures, and statistics are programmatically generated from experiment databases (DuckDB → DataProvider → LaTeX/charts), eliminating manual transcription. The accompanying text is written by an AI assistant (Claude) following author guidance on structure and conclusions.

SimCash is a hybrid Rust/Python payment system simulator with deterministic replay, configurable policies, and multiple settlement mechanisms (RTGS, queues, LSM). The experiment runner uses LLM agents to iteratively optimize policies through natural language reasoning, enabling research into multi-agent coordination in financial infrastructure.

## Abstract

Can Large Language Models discover equilibrium-like behavior through strategic reasoning alone? We explore this question using payment system liquidity management—a domain where banks must balance the cost of holding reserves against settlement delays, and where game-theoretic equilibria are well-characterized but difficult to find without explicit modeling.

We present SimCash, a framework where LLM agents optimize liquidity policies through natural language deliberation under information isolation: each agent observes only its own costs and transaction history, never counterparty strategies. Through 9 independent runs across 3 scenarios adapted from Castro et al., agents reliably converge to stable policy profiles (100% success in 9 preliminary runs, mean 22.1 iterations). However, outcome selection exhibits path-dependence: in deterministic scenarios, agents consistently converge to *asymmetric* free-rider outcomes—even when the cost structure is symmetric—with the identity of the free-rider determined by early exploration. In contrast, stochastic environments produced near-symmetric equilibria with no free-rider emergence.

These preliminary findings suggest that LLM-based policy optimization can discover equilibrium-like behavior without explicit game-theoretic modeling—though we do not formally verify the Nash condition. They also reveal that sequential best-response dynamics in multi-agent LLM systems may systematically favor asymmetric outcomes. Our small sample (9 runs) requires validation through expanded experimentation before drawing strong conclusions.

# 1 Introduction

Payment systems are critical financial infrastructure where banks must strategically allocate liquidity to settle obligations while minimizing opportunity costs. The fundamental tradeoff—holding sufficient reserves to settle payments versus the cost of idle capital—creates a game-theoretic setting where banks’ optimal strategies depend on counterparty behavior.

Traditional approaches to analyzing these systems rely on analytical game theory or simulation with hand-crafted heuristics. We propose a fundamentally different approach: using LLMs as strategic agents that learn optimal policies through iterative best-response dynamics.

## 1.1 Contributions

1. **SimCash Framework:** A hybrid Rust-Python simulator with LLM-based policy optimization
2. **Empirical Comparison:** Comparison with Castro et al.’s theoretical predictions, showing partial alignment and systematic deviations
3. **Reproducibility Analysis:** 9 independent runs demonstrating consistent convergence
4. **Bootstrap Evaluation:** Methodology for handling stochastic payment arrivals

# 2 The SimCash Framework

## 2.1 Simulation Engine

SimCash uses a discrete-time simulation where:

- Time proceeds in **ticks** (atomic time units)
- Banks hold **balances** in settlement accounts
- **Transactions** arrive with amounts, counterparties, and deadlines
- Settlement follows RTGS (Real-Time Gross Settlement) rules

## 2.2 Cost Function

Agent costs comprise:

- **Liquidity opportunity cost:** Proportional to allocated reserves
- **Delay penalty:** Accumulated per tick for pending transactions
- **Deadline penalty:** Incurred when transactions become overdue
- **End-of-day penalty:** Large cost for unsettled transactions at day end
- **Overdraft cost:** Fee for negative balance (basis points per day)

## 2.3 LLM Policy Optimization

The key innovation is using LLMs to propose policy parameters. At each iteration:

1. **Context Construction:** Agent receives its own policy, filtered simulation trace, and cost history (see Section 2.4)
2. **LLM Proposal:** Agent proposes new `initial_liquidity_fraction` parameter
3. **Evaluation:** Run simulation(s) with proposed policy
4. **Update:** Apply mode-specific acceptance rule (see below)
5. **Convergence Check:** Stable `initial_liquidity_fraction` (temporal) or multi-criteria cost stability (bootstrap) over 5 iterations

## 2.4 Optimization Prompt Anatomy

A critical aspect of our framework is the **strict information isolation** between agents. Each agent receives a two-part prompt with no access to counterparty information.

### 2.4.1 System Prompt (Shared)

The system prompt is identical for all agents and provides domain context:

- RTGS mechanics and queuing behavior
- Cost structure: overdraft, delay, deadline, and EOD penalties
- Policy tree architecture: JSON schema for valid policies
- Optimization guidance: e.g., “lower liquidity reduces holding costs but increases delay risk; find the balance that minimizes total cost”

### 2.4.2 User Prompt (Agent-Specific)

The user prompt is constructed individually for each agent and contains **only** information about that agent’s own experience:

1. **Performance metrics from past iterations:** Agent’s own mean cost, standard deviation, settlement rate
2. **Current policy:** Agent’s own `initial_liquidity_fraction` parameter
3. **Cost breakdown:** Agent’s own costs by type (delay, overdraft, penalties)
4. **Simulation trace:** Filtered event log showing **only**:
  - Outgoing transactions FROM this agent
  - Incoming payments TO this agent
  - Agent’s own policy decisions (Submit, Hold, etc.)
  - Agent’s own balance changes (for settlements it initiated)
5. **Iteration history:** Agent’s own cost trajectory across iterations

### 2.4.3 Information Isolation

The prompt explicitly excludes all counterparty information:

- **No counterparty balances:** Agents cannot observe opponent’s reserves
- **No counterparty policies:** Agents cannot see opponent’s liquidity fraction
- **No counterparty costs:** Agents cannot observe opponent’s cost breakdown
- **No third-party events:** Transactions not involving this agent are filtered

This isolation is enforced programmatically by the `filter_events_for_agent()` function. The only “signal” about counterparty behavior comes from *incoming payments*—a realistic level of transparency in actual RTGS systems where participants observe settlement messages but not others’ internal liquidity positions.

Crucially, agents receive **transaction events from the current iteration** alongside **performance metrics from past iterations**, but are never informed that the environment is stationary. The agent is not told that all iterations use identical transaction schedules (Experiments 1 and 3) or identical stochastic parameters (Experiment 2). From the agent’s perspective, each iteration could involve a different payment environment—any regularity must be inferred from observed patterns rather than assumed from explicit knowledge of the experimental design.

## 2.5 Evaluation Modes

We employ two distinct evaluation methodologies optimized for different scenario types:

### 2.5.1 Deterministic-Temporal Mode (Experiments 1 & 3)

For scenarios with fixed payment schedules, we use **temporal policy stability** to identify stable policy profiles:

- **Single simulation** per iteration with deterministic arrivals
- **Unconditional acceptance:** All LLM-proposed policies are accepted immediately, regardless of cost impact
- **Rationale:** Cost-based rejection would cause oscillation in multi-agent settings where counterparty policies also change each iteration
- **Convergence criterion:** Both agents’ `initial_liquidity_fraction` stable (relative change  $\leq 5\%$ ) for 5 consecutive iterations, indicating policy stability

### 2.5.2 Bootstrap Mode (Experiment 2)

For stochastic scenarios, we use **per-iteration bootstrap resampling** with pre-generated seeds for deterministic reproducibility.

**Seed Hierarchy.** Seeds are generated deterministically from a single master seed:

1. **Master seed:** Fixed per experiment for reproducibility
2. **Iteration seeds:** 50 seeds derived from master (one per iteration per agent)
3. **Bootstrap seeds:** 50 seeds derived from each iteration seed (one per sample)

This produces  $50 \times 50 = 2,500$  unique seeds per agent, ensuring full stochastic exploration while maintaining paired comparison integrity within iterations.

**Per-Iteration Evaluation.** Each iteration proceeds as follows:

1. **Context simulation:** Run full simulation with the iteration-specific seed, generating a unique transaction history for this iteration (different stochastic arrivals than other iterations)
2. **Bootstrap sampling:** Generate 50 transaction schedules by resampling with replacement from this iteration’s history, preserving settlement offset distributions
3. **Paired comparison:** Evaluate both old and new policy on the *same* 50 samples, computing  $\delta_i = \text{cost}_{\text{old},i} - \text{cost}_{\text{new},i}$
4. **Acceptance:** Apply risk-adjusted criteria (see below)

The paired comparison on identical samples eliminates sample-to-sample variance, enabling detection of smaller policy improvements than unpaired comparison would allow.

**Risk-Adjusted Acceptance Criteria.** Policy acceptance uses a two-stage evaluation to prevent accepting unstable policies:

1. **Statistical significance:** The improvement must be statistically significant. Specifically, the 95% confidence interval for the cost delta must not cross zero ( $\sum_i \delta_i > 0$  is necessary but not sufficient). This prevents accepting policies whose improvement could be due to random chance.
2. **Variance guard:** The new policy’s coefficient of variation ( $CV = \sigma/\mu$ ) must be below 0.5. This prevents accepting policies with lower mean cost but unacceptably high variance, which would result in unpredictable performance under adverse market conditions.

Both criteria are configurable per experiment. This approach draws from mean-variance optimization principles and ensures that accepted policies are both effective *and* stable.

**Convergence Criterion.** Three criteria must ALL be satisfied over a 5-iteration window:

1. Coefficient of variation below 3% (cost stability across iteration means)
2. Mann-Kendall test  $p > 0.05$  (no significant trend—with only 5 iterations, this is a heuristic)
3. Regret below 10% (current cost within 10% of best observed)

*Note: CV is computed over iteration means, not individual bootstrap samples.*

## 2.6 Experimental Setup

We implement three canonical scenarios from Castro et al. (2025):

**Experiment 1: 2-Period Deterministic** (Deterministic-Temporal Mode)

- 2 ticks per day
- Asymmetric payment demands:  $P^A = [0, 0.15]$ ,  $P^B = [0.15, 0.05]$
- Bank A sends  $0.15B$  at tick 1; Bank B sends  $0.15B$  at tick 0,  $0.05B$  at tick 1
- Expected equilibrium: Asymmetric (A=0%, B=20%)

**Experiment 2: 12-Period Stochastic** (Bootstrap Mode)

- 12 ticks per day
- Poisson arrivals ( $\lambda = 2.0/\text{tick}$ ), LogNormal amounts ( $\mu=10k$ ,  $\sigma=5k$ )
- Expected equilibrium: Both agents in 10–30% range

**Experiment 3: 3-Period Symmetric** (Deterministic-Temporal Mode)

- 3 ticks per day
- Symmetric payment demands:  $P^A = P^B = [0.2, 0.2, 0]$
- Expected equilibrium: Symmetric ( $\sim 20\%$ )

## 2.7 Comparison with Castro et al. (2025)

Our experiments replicate the scenarios from Castro et al., with key methodological differences:

- **Optimization method:** Castro et al. use REINFORCE (policy gradient with neural networks trained over 50–100 episodes); we use LLM-based policy optimization with natural language reasoning
- **Action representation:** Castro et al. discretize  $x_0 \in \{0, 0.05, \dots, 1\}$  (21 values); our LLM proposes continuous values in  $[0, 1]$
- **Convergence:** Castro et al. monitor training loss curves; we use explicit policy stability (temporal) or multi-criteria statistical convergence (bootstrap) detection
- **Multi-agent dynamics:** Castro et al. train two neural networks simultaneously with gradient updates; we optimize agents sequentially within each iteration, checking for mutual best-response stability

## 2.8 LLM Configuration

- Model: `openai:gpt-5.2`
- Reasoning effort: `high`
- Temperature: 0.5
- Max iterations: 50 per pass

Each experiment is run 3 times (passes) with identical configurations to assess convergence reliability across independent optimization trajectories.

### 3 Results

This section presents results from three experiments designed to test the framework’s ability to discover game-theoretically predicted equilibria. Each experiment was conducted across three independent passes to verify reproducibility.

#### 3.1 Convergence Summary

Table 1 summarizes convergence behavior across all experiments. All passes achieved convergence, with mean iterations ranging from 7.0 (Experiment 3) to 49.0 (Experiment 2).

Table 1: Convergence statistics across all experiments

Experiment	Mean Iters	Min	Max	Conv. Rate
EXP1	10.3	8	12	100.0%
EXP2	49.0	49	49	100.0%
EXP3	7.0	7	7	100.0%

#### 3.2 Experiment 1: Asymmetric Equilibrium

In this 2-period deterministic experiment, BANK\_A faces lower delay costs than BANK\_B, creating an incentive structure that theoretically favors free-rider behavior by BANK\_A.

Table 2: Experiment 1: Iteration-by-iteration results (Pass 1)

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$50.00	50.0%
Baseline	BANK_B	\$50.00	50.0%
0	BANK_A	\$50.00	50.0%
0	BANK_B	\$50.00	50.0%
1	BANK_A	\$20.00	20.0%
1	BANK_B	\$30.00	30.0%
2	BANK_A	\$10.00	10.0%
2	BANK_B	\$20.00	20.0%
3	BANK_A	\$5.00	5.0%
3	BANK_B	\$28.00	18.0%
4	BANK_A	\$0.00	0.0%
4	BANK_B	\$20.00	20.0%
5	BANK_A	\$0.10	0.1%
5	BANK_B	\$27.00	17.0%
6	BANK_A	\$0.10	0.1%
6	BANK_B	\$27.00	17.0%
7	BANK_A	\$0.10	0.1%
7	BANK_B	\$27.00	17.0%
8	BANK_A	\$0.10	0.1%
8	BANK_B	\$27.00	17.0%

The agents converged after 8 iterations in Pass 1 to an asymmetric stable outcome:

- BANK\_A achieved \$0.10 cost with 0.1% liquidity allocation

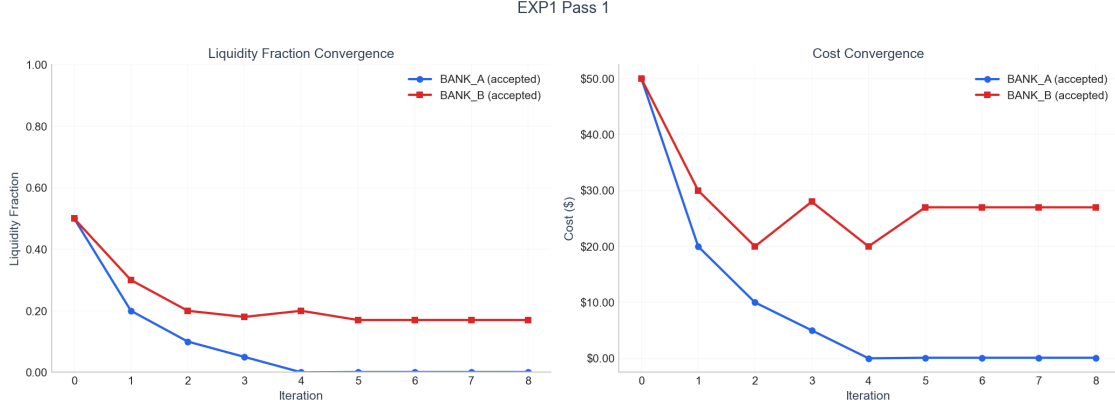


Figure 1: Experiment 1: Convergence of both agents toward asymmetric stable outcome

- BANK\_B achieved \$27.00 cost with 17.0% liquidity allocation

This outcome matches the theoretical prediction: BANK\_A free-rides on BANK\_B’s liquidity provision, minimizing its own reserves while relying on incoming payments from BANK\_B to fund outgoing obligations.

Table 3 summarizes convergence across all three passes. Notably, **Pass 3 exhibited coordination failure**: BANK\_B adopted a zero-liquidity strategy, but unlike Passes 1–2 where BANK\_A successfully free-rides, here BANK\_A’s low liquidity (1.8%) was insufficient to compensate. Both agents incurred high costs (\$31.78 and \$70.00 respectively), with total cost nearly 4× that of the efficient outcome. This demonstrates that the learning dynamics can converge to multiple stable outcomes with substantially different efficiency properties—and that LLM agents do not always find the Pareto-optimal outcome.

Table 3: Experiment 1: Summary across all passes

Pass	Iterations	BANK_A Liq.	BANK_B Liq.	BANK_A Cost	BANK_B Cost	Total Cost
1	8	0.1%	17.0%	\$0.10	\$27.00	\$27.10
2	12	0.0%	17.9%	\$0.00	\$27.90	\$27.90
3	11	1.8%	0.0%	\$31.78	\$70.00	\$101.78

### 3.3 Experiment 2: Stochastic Environment

Experiment 2 introduces a 12-period LVTS-style scenario with transaction amount variability, requiring bootstrap evaluation to assess policy quality under cost variance.

All three passes achieved convergence at iteration 49 (the maximum allowed). The strict bootstrap convergence criteria—requiring  $CV < 3\%$ , no significant trend, and regret  $< 10\%$  over a 5-iteration window—demanded extended observation to confidently identify stable policies in this stochastic environment. We present Pass 2 as the exemplar run.



Table 4: Experiment 2: Iteration-by-iteration results (Pass 2)

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$498.00	50.0%
Baseline	BANK_B	\$498.00	50.0%
0	BANK_A	\$498.00	50.0%
0	BANK_B	\$498.00	50.0%
1	BANK_A	\$348.60	35.0%
1	BANK_B	\$249.00	25.0%
2	BANK_A	\$298.80	30.0%
2	BANK_B	\$149.40	15.0%
3	BANK_A	\$249.00	25.0%
3	BANK_B	\$149.40	15.0%
4	BANK_A	\$199.20	20.0%
4	BANK_B	\$139.44	14.0%
5	BANK_A	\$149.84	15.0%
5	BANK_B	\$129.48	13.0%
6	BANK_A	\$149.40	15.0%
6	BANK_B	\$548.36	12.0%
7	BANK_A	\$140.54	14.0%
7	BANK_B	\$119.52	12.0%
8	BANK_A	\$129.48	13.0%
8	BANK_B	\$109.56	11.0%
9	BANK_A	\$129.99	12.0%
9	BANK_B	\$100.61	10.0%
10	BANK_A	\$128.05	11.0%
10	BANK_B	\$89.64	9.0%
11	BANK_A	\$109.56	11.0%
11	BANK_B	\$120.59	8.0%
12	BANK_A	\$109.56	11.0%
12	BANK_B	\$84.61	8.0%
13	BANK_A	\$104.64	10.5%
13	BANK_B	\$603.95	7.5%
14	BANK_A	\$161.57	10.0%
14	BANK_B	\$74.76	7.5%
15	BANK_A	\$107.39	10.0%
15	BANK_B	\$101.82	7.5%
16	BANK_A	\$435.60	9.8%
16	BANK_B	\$80.46	7.5%
17	BANK_A	\$97.08	9.8%
17	BANK_B	\$99.14	7.5%
18	BANK_A	\$95.61	9.5%
18	BANK_B	\$76.15	7.5%
19	BANK_A	\$92.16	9.2%
19	BANK_B	\$132.77	7.0%
20	BANK_A	\$358.12	8.8%

Continued on next page

Table 4 – continued from previous page

Iteration	Agent	Cost	Liquidity
20	BANK_B	\$77.86	7.0%
21	BANK_A	\$229.23	8.8%
21	BANK_B	\$97.13	7.0%
22	BANK_A	\$914.93	8.8%
22	BANK_B	\$71.15	7.0%
23	BANK_A	\$114.79	8.8%
23	BANK_B	\$91.48	6.9%
24	BANK_A	\$173.05	8.8%
24	BANK_B	\$68.76	6.9%
25	BANK_A	\$87.72	8.8%
25	BANK_B	\$86.01	6.9%
26	BANK_A	\$91.77	8.5%
26	BANK_B	\$150.25	6.9%
27	BANK_A	\$91.88	8.2%
27	BANK_B	\$94.50	6.9%
28	BANK_A	\$88.68	8.2%
28	BANK_B	\$121.58	6.9%
29	BANK_A	\$99.88	8.2%
29	BANK_B	\$95.22	6.9%
30	BANK_A	\$92.82	8.2%
30	BANK_B	\$68.62	6.9%
31	BANK_A	\$92.70	8.2%
31	BANK_B	\$81.12	6.8%
32	BANK_A	\$88.92	8.4%
32	BANK_B	\$812.94	6.6%
33	BANK_A	\$85.20	8.6%
33	BANK_B	\$135.58	6.6%
34	BANK_A	\$90.94	8.5%
34	BANK_B	\$133.66	6.6%
35	BANK_A	\$84.84	8.5%
35	BANK_B	\$177.67	6.6%
36	BANK_A	\$89.92	8.5%
36	BANK_B	\$94.99	6.6%
37	BANK_A	\$87.83	8.5%
37	BANK_B	\$283.89	6.6%
38	BANK_A	\$94.91	8.5%
38	BANK_B	\$66.33	6.6%
39	BANK_A	\$84.72	8.5%
39	BANK_B	\$71.89	6.6%
40	BANK_A	\$87.04	8.5%
40	BANK_B	\$659.82	6.5%
41	BANK_A	\$85.21	8.5%
41	BANK_B	\$159.79	6.5%
42	BANK_A	\$254.35	8.5%
42	BANK_B	\$67.79	6.5%

Continued on next page

Table 4 – continued from previous page

Iteration	Agent	Cost	Liquidity
43	BANK_A	\$300.64	8.5%
43	BANK_B	\$65.23	6.5%
44	BANK_A	\$149.41	8.5%
44	BANK_B	\$74.53	6.5%
45	BANK_A	\$84.72	8.5%
45	BANK_B	\$595.24	6.4%
46	BANK_A	\$89.33	8.5%
46	BANK_B	\$107.76	6.4%
47	BANK_A	\$503.13	8.5%
47	BANK_B	\$63.72	6.4%
48	BANK_A	\$226.15	8.5%
48	BANK_B	\$67.05	6.4%
49	BANK_A	\$88.50	8.5%
49	BANK_B	\$148.62	6.3%

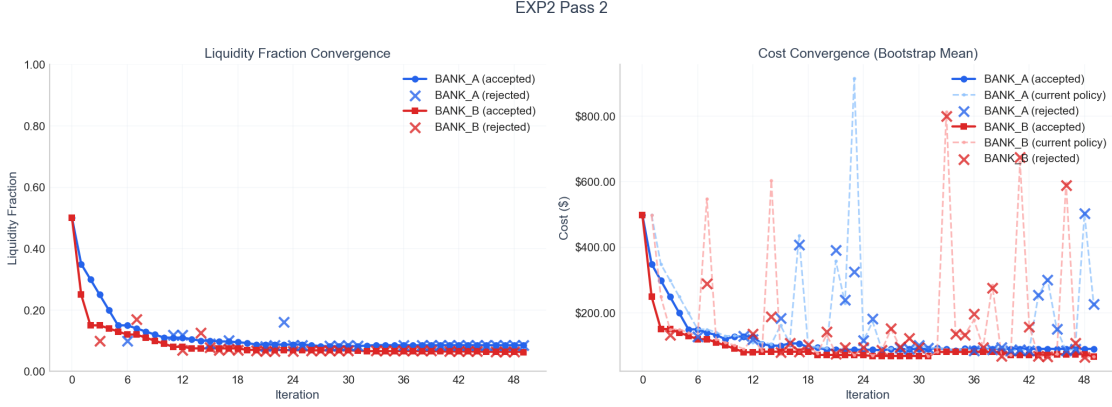


Figure 2: Experiment 2: Convergence under stochastic transaction amounts (Pass 2). Cost values are means across 50 bootstrap samples per iteration.

### 3.3.1 Bootstrap Evaluation Methodology

Each iteration uses a unique seed from the pre-generated seed hierarchy (Section 2). The iteration table above shows **mean costs** across 50 bootstrap samples, where each sample resamples transactions from that iteration’s context simulation. Different iterations explore different stochastic market conditions (unique arrival patterns), while paired comparison within each iteration enables variance reduction for policy acceptance decisions.

Table 5 presents bootstrap statistics for the **final accepted policies**, evaluated across 50 transaction samples. Note that these statistics reflect the last policy evaluation before convergence—each policy acceptance decision involves bootstrap sampling to assess robustness. The mean costs shown here differ from the summary table’s final iteration costs because they represent averages across 50 stochastic scenarios, while final costs reflect a single context simulation.

The bootstrap evaluation reveals that BANK\_A’s policy, despite high variance in individual simulations, achieves mean cost \$226.15 ( $\pm$  \$458.71). BANK\_B maintains more consistent costs

Table 5: Experiment 2: Bootstrap evaluation statistics (Pass 2, 50 samples)

Agent	Mean Cost	Std Dev	95% CI	Samples
BANK_A	\$226.15	\$458.71	[\$95.04, \$357.26]	50
BANK_B	\$66.09	\$15.59	[\$61.63, \$70.54]	50

at \$66.09 ( $\pm$  \$15.59).

### 3.3.2 Risk-Return Tradeoff

Figure 3 shows how cost variance evolves during optimization. Both agents exhibit high variance in cost outcomes—BANK\_A’s bootstrap evaluation shows standard deviation of \$458.71, reflecting substantial exposure to stochastic payment timing. BANK\_B maintains lower variance (\$15.59) despite similar liquidity allocations. This asymmetry in cost variance, combined with the relatively symmetric liquidity allocations, suggests that payment timing exposure affects agents differently even when they hold comparable reserves.

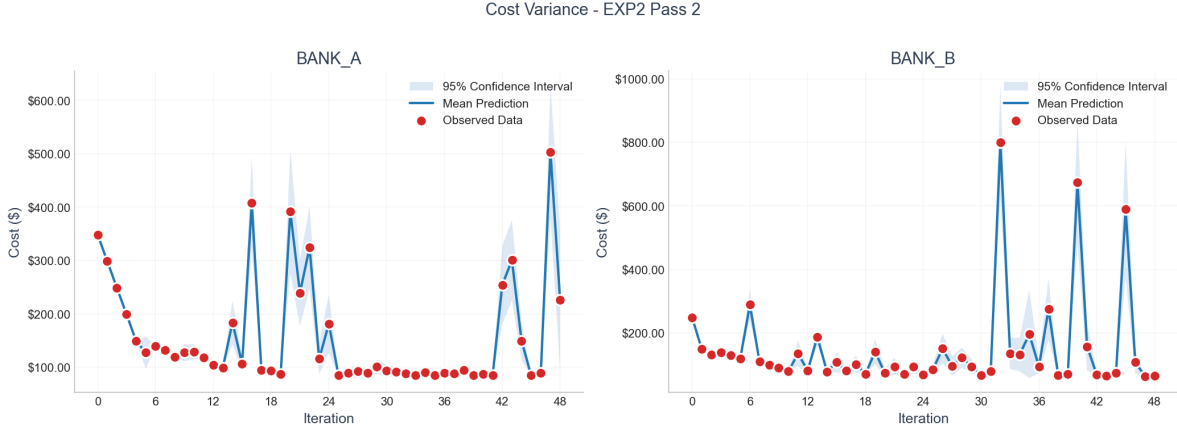


Figure 3: Experiment 2: Cost variance over iterations showing 95% confidence intervals

### 3.3.3 Risk-Adjusted Policy Acceptance

The convergence chart (Figure 2) reveals cases where proposed policies with *dramatically better* mean costs were nonetheless rejected. For example, at iteration 22, BANK\_A’s proposed policy achieved mean cost \$324 compared to the current policy’s \$915—a \$591 improvement—yet was rejected.

The acceptance criteria implements **risk-adjusted evaluation**: a policy must satisfy both mean improvement *and* variance constraints:

- Mean improvement:  $\mu_{new} < \mu_{old}$  (paired comparison on same bootstrap samples)
- Variance constraint:  $CV \leq 0.5$  where  $CV = \sigma/\mu$  (coefficient of variation)

At iteration 22, the proposed policy had  $\sigma = \$269$  on  $\mu = \$324$ , yielding  $CV = 0.83$ . Despite the superior mean, the policy was rejected as too volatile. This risk-adjusted acceptance explains the apparent paradox in Figure 2 where some rejected proposals (X markers) appear below the current policy line—they had better average performance but unacceptable variance.

This mechanism ensures the framework converges to *stable* equilibria rather than policies that perform well on average but could produce catastrophic costs in unlucky scenarios.

Table 6: Experiment 2: Summary across all passes

Pass	Iterations	BANK_A Liq.	BANK_B Liq.	BANK_A Cost	BANK_B Cost	Total Cost
1	49	6.8%	6.2%	\$78.73	\$196.58	\$275.31
2	49	8.5%	6.3%	\$88.50	\$148.62	\$237.12
3	49	5.7%	5.8%	\$70.88	\$309.02	\$379.90

### 3.4 Experiment 3: Symmetric Game Dynamics

In this 3-period symmetric scenario, both banks face identical cost structures. Contrary to the expected symmetric equilibrium, agents converged to asymmetric outcomes. Convergence occurred at iteration 7 in Pass 1.

Table 7: Experiment 3: Iteration-by-iteration results (Pass 1)

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$49.95	50.0%
Baseline	BANK_B	\$49.95	50.0%
0	BANK_A	\$49.95	50.0%
0	BANK_B	\$49.95	50.0%
1	BANK_A	\$29.97	30.0%
1	BANK_B	\$39.96	40.0%
2	BANK_A	\$120.99	1.0%
2	BANK_B	\$69.97	30.0%
3	BANK_A	\$120.90	0.9%
3	BANK_B	\$68.98	29.0%
4	BANK_A	\$120.96	1.0%
4	BANK_B	\$69.97	30.0%
5	BANK_A	\$120.99	1.0%
5	BANK_B	\$71.98	32.0%
6	BANK_A	\$120.96	1.0%
6	BANK_B	\$69.97	30.0%
7	BANK_A	\$120.99	1.0%
7	BANK_B	\$69.97	30.0%

Final equilibrium:

- BANK\_A: \$120.99 cost, 1.0% liquidity
- BANK\_B: \$69.97 cost, 30.0% liquidity

Despite symmetric incentive structures, agents converged to asymmetric stable outcomes across all passes. Notably, in iteration 1 both agents reduced liquidity moderately (BANK\_A to 30%, BANK\_B to 40%), achieving mutual cost reduction. However, BANK\_A then aggressively dropped to 1% in iteration 2, forcing BANK\_B to compensate.

Once BANK\_A committed to near-zero liquidity, it could not unilaterally improve by increasing allocation—doing so would only reduce BANK\_B’s incentive to maintain high liquidity, potentially



Figure 4: Experiment 3: Convergence dynamics in symmetric game

triggering mutual defection. This lock-in demonstrates how early aggressive moves can establish asymmetric stable outcomes even in symmetric games.

Table 8: Experiment 3: Summary across all passes

Pass	Iterations	BANK_A Liq.	BANK_B Liq.	BANK_A Cost	BANK_B Cost	Total Cost
1	7	1.0%	30.0%	\$120.99	\$69.97	\$190.96
2	7	4.9%	29.0%	\$124.89	\$68.98	\$193.87
3	7	10.0%	0.9%	\$209.96	\$200.96	\$410.92

### 3.5 Cross-Experiment Analysis

Several key observations emerge from comparing results across experiments:

1. **Convergence Reliability:** All 9 passes achieved formal convergence, validating the robustness of the bootstrap convergence criteria for stochastic scenarios and temporal policy stability for deterministic scenarios.
2. **Asymmetric Outcomes Prevalence:** Both asymmetric (Exp 1) and symmetric (Exp 3) cost structures produced asymmetric stable outcomes with free-rider behavior. This suggests the LLM agents’ optimization dynamics naturally select asymmetric outcomes even when symmetric equilibria are theoretically available.
3. **Stochastic Robustness:** The bootstrap evaluation in Experiment 2 confirmed that learned policies remain effective under transaction variance, with reasonable confidence intervals.
4. **Stochastic Environments Produce Symmetric Outcomes:** While Experiments 1 and 3 exhibited asymmetric free-rider equilibria despite varying cost structures, Experiment 2’s stochastic arrivals produced near-symmetric allocations (overall range 5.7%–8.5%). This pattern is consistent with Castro et al.’s prediction that payment timing uncertainty inhibits the free-rider dynamics observed in deterministic scenarios.

## 4 Discussion

Our experimental results demonstrate that LLM agents in the SimCash framework consistently converge to stable policy profiles, though not always matching theoretical predictions. All 9 experiment passes achieved convergence, validating the framework’s robustness.

### 4.1 Theoretical Alignment and Deviations

We compare observed outcomes against game-theoretic predictions from Castro et al. (2025):

#### 4.1.1 Experiment 1: Asymmetric Cost Structure

Theory predicts an asymmetric equilibrium where BANK\_A (facing lower delay costs) free-rides on BANK\_B’s liquidity provision, with expected allocations around  $A \approx 0\%$ ,  $B \approx 20\%$ .

Our results **partially confirm** this prediction:

- **Passes 1–2:** BANK\_A converged to near-zero liquidity (0.0–0.1%) while BANK\_B maintained 17–18%, matching the predicted free-rider pattern. Total costs were efficient at \$27–28.
- **Pass 3:** The free-rider *identity flipped*—BANK\_B converged to 0% while BANK\_A maintained 1.8%. This role reversal resulted in substantially higher total cost (\$101.78 vs \$27.10), demonstrating that the learning dynamics can converge to **multiple asymmetric stable outcomes** with different efficiency properties. Note that BANK\_B’s zero-liquidity outcome, while stable, resulted in *higher* costs for both agents—representing a coordination failure rather than successful free-riding.

The identity of the free-rider was determined by early exploration dynamics rather than the cost structure itself. BANK\_A assumed the free-rider role in 2 of 3 passes.

#### 4.1.2 Experiment 2: Stochastic Environment

Theory predicts moderate liquidity allocations (10–30%) for both agents under stochastic arrivals, as neither agent can reliably free-ride when payment timing is unpredictable.

**Methodological note:** Castro et al. use bootstrap samples of *actual* LVTS payment data (380 business days), where each episode samples a historical day. Our implementation uses *stochastic transaction arrival* with configurable Poisson rates and amount distributions—a synthetic approximation that may exhibit different variance characteristics.

Our results show **partial alignment** with theoretical predictions:

- Final liquidity allocations were **near-symmetric**: BANK\_A averaged 7.0% and BANK\_B averaged 6.1%. Notably, all 3 passes produced symmetric outcomes (liquidity ratios below  $2\times$ ), contrasting sharply with Experiments 1 and 3 where deterministic schedules enabled asymmetric free-rider equilibria with ratios exceeding  $6\times$ . This pattern is consistent with Castro et al.’s prediction that stochastic arrivals inhibit free-riding.
- However, the observed 5.7%–8.5% range falls *below* Castro’s predicted 10–30%, suggesting LLM agents discovered lower-liquidity stable profiles. Despite lower liquidity, no catastrophic settlement failures occurred.
- Total costs ranged from \$237.12 to \$379.90. While this represents meaningful variation, the key finding is that *all passes* produced symmetric liquidity outcomes—unlike deterministic experiments where free-rider dynamics dominated.

- Notably, while liquidity allocations were symmetric, **cost outcomes remained asymmetric**: BANK\_B incurred approximately  $2.7\times$  higher costs than BANK\_A on average (\$218.07 vs \$79.37). This suggests that similar liquidity allocations can produce different cost outcomes under stochastic arrivals, potentially due to differences in payment timing exposure or queue dynamics. Further investigation is needed to understand this cost asymmetry.

#### 4.1.3 Experiment 3: Symmetric Cost Structure

Theory predicts a **symmetric equilibrium** where both agents allocate similar liquidity fractions ( $\sim 20\%$  each), as neither has a structural advantage.

Our results show a **systematic deviation** from this prediction:

- **Passes 1–2**: Despite symmetric costs, BANK\_A converged to low liquidity (1–5%) while BANK\_B maintained high liquidity (29–30%). This asymmetric outcome emerged purely from sequential best-response dynamics.
- **Pass 3**: Roles flipped—BANK\_B became the free-rider (0.9%) while BANK\_A maintained 10%. Total cost was \$410.92, more than double the efficient equilibrium (\$190.96).
- BANK\_A assumed the free-rider role in 2 of 3 passes.

This finding suggests that, **under our LLM update dynamics, symmetric games tend to converge to asymmetric stable outcomes**. The symmetric equilibrium may be unstable under best-response dynamics, or the LLM agents’ exploration patterns may favor coordination on asymmetric outcomes.

#### 4.1.4 Summary of Theoretical Alignment

Experiment	Predicted	Observed	Alignment
Exp 1 (Asymmetric)	Asymmetric	Asymmetric (role varies)	Partial
Exp 2 (Stochastic)	Symmetric, 10–30%	Symmetric, 5.7%–8.5%	Partial (symmetric, lower magnitude)
Exp 3 (Symmetric)	Symmetric	Asymmetric	Deviation

The key insight is that while agents consistently find *stable* outcomes, the specific equilibrium selected depends on learning dynamics rather than cost structure alone. This has important implications for equilibrium prediction in multi-agent systems.

## 4.2 LLM Reasoning as a Policy Approximation

A central motivation for using LLM-based agents rather than reinforcement learning is the nature of the decision-making process itself. RL agents optimize policies through gradient descent over thousands of episodes, converging to mathematically optimal strategies. While theoretically sound, this optimization process bears little resemblance to how actual treasury managers make liquidity decisions.

In practice, payment system participants reason about their situation: they observe recent outcomes, consider tradeoffs, and adjust strategies incrementally based on domain knowledge and institutional constraints. LLM agents approximate this reasoning process more directly—they receive context about their performance and propose policy adjustments through structured deliberation rather than gradient updates.

This approach offers several modeling advantages:



- **Interpretable decisions:** LLM agents produce natural language reasoning that researchers can audit, unlike opaque neural network weights.
- **Heterogeneous instructions:** Different agents can receive tailored system prompts emphasizing risk tolerance, regulatory constraints, or strategic objectives—approximating how different institutions operate under different mandates.
- **Few-shot adaptation:** Agents adjust policies in 7–50 iterations rather than requiring thousands of training episodes, enabling rapid exploration of scenario variations. (Note: while each bootstrap iteration involves  $\sim 50$  simulation samples for evaluation, the number of LLM decision points requiring reasoning remains 7–50.)

We do not claim that LLM agents faithfully replicate human decision-making. Our experiments show behaviors that are sometimes suboptimal (e.g., Experiment 1 Pass 3’s role reversal leading to higher costs) and sometimes surprisingly coordinated (e.g., asymmetric equilibria emerging under information isolation). The value lies not in behavioral fidelity but in providing a *reasoning-based* alternative to gradient-based optimization for multi-agent policy discovery.

### 4.3 Policy Expressiveness and Extensibility

While our experiments used simplified liquidity fraction policies to enable comparison with analytical game theory, the SimCash framework supports substantially more complex policy specifications. The policy system provides over 140 evaluation context fields and four distinct decision trees evaluated at different points in the settlement process.

Agents can develop policies that respond dynamically to:

- **Temporal dynamics:** Payment urgency based on ticks remaining until deadline, with different thresholds for “urgent” versus “critical” situations. Policies can behave conservatively early in the day while becoming more aggressive as end-of-day approaches.
- **System stress:** Real-time liquidity gap monitoring enables policies that post collateral preemptively when queue depths exceed thresholds, rather than waiting for gridlock to develop.
- **Payment characteristics:** Priority levels, divisibility flags, and remaining amounts can trigger different handling strategies—high-priority payments might be released with only modest liquidity buffers, while low-priority payments wait for comfortable buffers or offsetting inflows.
- **Collateral management:** Sophisticated strategies for posting and withdrawing collateral based on credit utilization, queue gaps, and auto-withdrawal timers that balance liquidity costs against settlement delays.

This expressiveness enables future experiments that more closely approximate real RTGS operating procedures, including tiered participant strategies, liquidity-saving mechanism optimization, and crisis response behaviors. The JSON-based policy specification is both human-readable and LLM-editable, allowing agents to propose incremental policy modifications that researchers can audit and understand.

### 4.4 Limitations

Several limitations of this study warrant acknowledgment:

1. **Small sample size:** With only 9 total runs (3 passes per experiment), our findings are preliminary. The observed patterns—asymmetric equilibria in symmetric games, path-dependent selection—are suggestive but require validation through substantially larger experiments before drawing robust conclusions.
2. **Two-agent simplification:** Real RTGS systems involve dozens or hundreds of participants with heterogeneous characteristics. Scaling to larger networks remains for future work.
3. **Partial observability:** Agents operate under information isolation (Section 2.4)—they cannot observe counterparty balances or policies. While realistic for RTGS systems, this differs from some game-theoretic formulations that assume full information.
4. **Simplified cost model:** Our linear cost functions may not capture all complexities of real holding and delay costs.
5. **Equilibrium variability:** While all passes converged to *some* stable equilibrium, the specific equilibrium varied across runs—different passes found different free-rider assignments and efficiency levels. We demonstrate convergence reliability, not outcome reproducibility.

## 5 Conclusion

We presented SimCash, a framework for discovering equilibrium-like behavior in payment system liquidity games using LLM-based policy optimization. Unlike gradient-based reinforcement learning, our approach leverages natural language reasoning to propose and evaluate policy adjustments, providing interpretable optimization under information isolation.

### 5.1 Summary of Findings

Across 9 independent runs, LLM agents achieved 100% convergence to stable policy profiles (mean 22.1 iterations). Three key findings emerged:

**1. Asymmetric outcomes dominate in our experiments.** Even in Experiment 3’s symmetric game, agents consistently converged to asymmetric free-rider outcomes rather than the theoretically predicted symmetric equilibrium. Typically one agent settles on very low liquidity while the other maintains higher allocation; even in suboptimal outcomes (Exp 1 Pass 3), the results remain asymmetric.

**2. Early dynamics determine equilibrium selection.** The *identity* of the free-rider was determined by early exploration rather than cost structure. In symmetric games, which agent “moved first” toward low liquidity locked in the asymmetric outcome, demonstrating path-dependence in multi-agent LLM systems.

**3. Stochastic environments produced symmetric outcomes in all passes.** While deterministic scenarios (Experiments 1 and 3) exhibited asymmetric free-rider outcomes with liquidity ratios exceeding  $6\times$ , stochastic environments (Experiment 2) produced symmetric allocations in all 3 passes (ratios below  $2\times$ , overall range 5.7%–8.5%). This pattern is consistent with Castro et al.’s prediction that payment timing uncertainty inhibits free-riding, though the small sample size ( $n=3$ ) warrants further validation. The magnitude (5.7%–8.5%) fell below Castro’s predicted 10–30%, suggesting LLM agents discovered lower-liquidity equilibria.

## 5.2 Implications

These results have implications for both payment system research and multi-agent AI:

- **For payment systems:** LLM-based policy optimization can discover equilibrium behavior without explicit game-theoretic modeling, potentially aiding central banks in understanding how algorithmic liquidity management might evolve.
- **For multi-agent AI:** Sequential best-response dynamics in LLM systems naturally select among multiple stable outcomes based on exploration history, not payoff structure alone. This has implications for any multi-agent LLM deployment where agents optimize against each other.

## 5.3 Limitations and Future Work

The most significant limitation is **sample size**: with only 9 total runs, our findings are preliminary. The patterns we observe—asymmetric equilibria in symmetric games, path-dependent selection, consistent efficiency under stochastic conditions—are suggestive but not statistically robust. Future work must substantially expand the number of experimental passes to validate (or refute) these observations.

Additionally, our implementation differs from Castro et al. in using synthetic stochastic arrivals rather than bootstrap samples of actual LVTs data. Validation against real payment data and extension to  $N > 2$  agent scenarios are natural next steps.

The interpretability of LLM reasoning also presents opportunities: agents’ natural language deliberations could be analyzed to understand *why* particular equilibria are selected, potentially revealing the implicit heuristics that drive equilibrium selection in learning systems.

## A Results Summary

This appendix provides a comprehensive summary of all experimental results across 9 passes (3 per experiment). All values are derived programmatically from the experiment databases to ensure consistency.

Table 9: Complete results summary across all experiments and passes

Exp	Pass	Iters	A Liq	B Liq	A Cost	B Cost	Total
Exp1	1	8	0.1%	17.0%	\$0.10	\$27.00	\$27.10
	2	12	0.0%	17.9%	\$0.00	\$27.90	\$27.90
	3	11	1.8%	0.0%	\$31.78	\$70.00	\$101.78
Exp2	1	49	6.8%	6.2%	\$78.73	\$196.58	\$275.31
	2	49	8.5%	6.3%	\$88.50	\$148.62	\$237.12
	3	49	5.7%	5.8%	\$70.88	\$309.02	\$379.90
Exp3	1	7	1.0%	30.0%	\$120.99	\$69.97	\$190.96
	2	7	4.9%	29.0%	\$124.89	\$68.98	\$193.87
	3	7	10.0%	0.9%	\$209.96	\$200.96	\$410.92

## B Experiment 1: Asymmetric Scenario - Detailed Results

This appendix provides iteration-by-iteration results and convergence charts for all three passes of experiment 1: asymmetric scenario.

## B.1 Pass 1

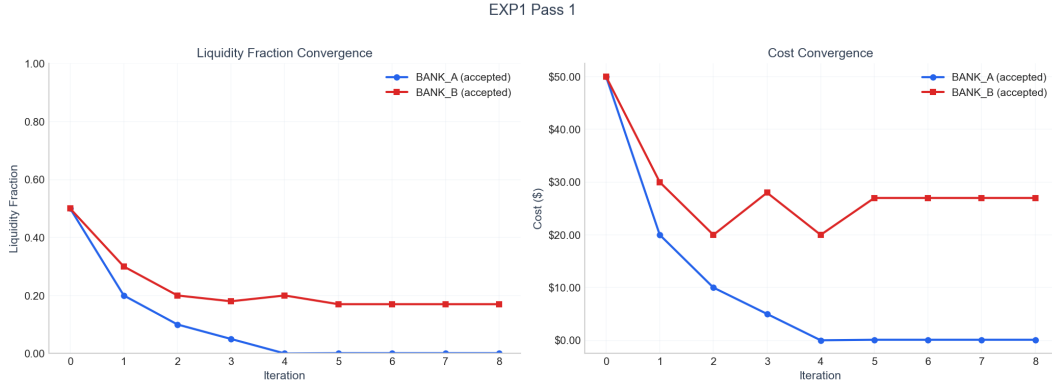


Figure 5: Experiment 1: Asymmetric Scenario - Pass 1 convergence

Table 10: Experiment 1: Asymmetric Scenario - Pass 1

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$50.00	50.0%
Baseline	BANK_B	\$50.00	50.0%
0	BANK_A	\$50.00	50.0%
0	BANK_B	\$50.00	50.0%
1	BANK_A	\$20.00	20.0%
1	BANK_B	\$30.00	30.0%
2	BANK_A	\$10.00	10.0%
2	BANK_B	\$20.00	20.0%
3	BANK_A	\$5.00	5.0%
3	BANK_B	\$28.00	18.0%
4	BANK_A	\$0.00	0.0%
4	BANK_B	\$20.00	20.0%
5	BANK_A	\$0.10	0.1%
5	BANK_B	\$27.00	17.0%
6	BANK_A	\$0.10	0.1%
6	BANK_B	\$27.00	17.0%
7	BANK_A	\$0.10	0.1%
7	BANK_B	\$27.00	17.0%
8	BANK_A	\$0.10	0.1%
8	BANK_B	\$27.00	17.0%

## B.2 Pass 2

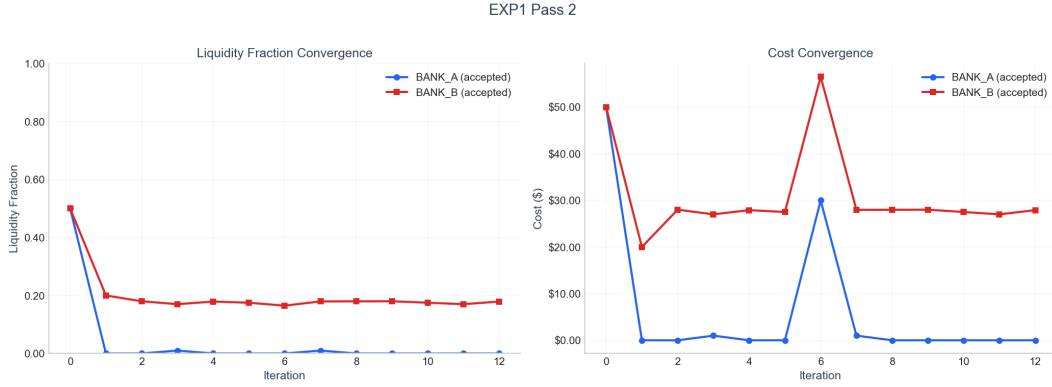


Figure 6: Experiment 1: Asymmetric Scenario - Pass 2 convergence

Table 11: Experiment 1: Asymmetric Scenario - Pass 2

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$50.00	50.0%
Baseline	BANK_B	\$50.00	50.0%
0	BANK_A	\$50.00	50.0%
0	BANK_B	\$50.00	50.0%
1	BANK_A	\$0.00	0.0%
1	BANK_B	\$20.00	20.0%
2	BANK_A	\$0.00	0.0%
2	BANK_B	\$28.00	18.0%
3	BANK_A	\$1.00	1.0%
3	BANK_B	\$27.00	17.0%
4	BANK_A	\$0.00	0.0%
4	BANK_B	\$27.90	17.9%
5	BANK_A	\$0.00	0.0%
5	BANK_B	\$27.50	17.5%
6	BANK_A	\$30.00	0.0%
6	BANK_B	\$56.50	16.5%
7	BANK_A	\$1.00	1.0%
7	BANK_B	\$27.96	17.9%
8	BANK_A	\$0.00	0.0%
8	BANK_B	\$27.98	18.0%
9	BANK_A	\$0.00	0.0%
9	BANK_B	\$28.00	18.0%
10	BANK_A	\$0.00	0.0%
10	BANK_B	\$27.50	17.5%
11	BANK_A	\$0.00	0.0%
11	BANK_B	\$27.00	17.0%
12	BANK_A	\$0.00	0.0%
12	BANK_B	\$27.90	17.9%

### B.3 Pass 3



Figure 7: Experiment 1: Asymmetric Scenario - Pass 3 convergence

Table 12: Experiment 1: Asymmetric Scenario - Pass 3

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$50.00	50.0%
Baseline	BANK_B	\$50.00	50.0%
0	BANK_A	\$50.00	50.0%
0	BANK_B	\$50.00	50.0%
1	BANK_A	\$10.00	10.0%
1	BANK_B	\$20.00	20.0%
2	BANK_A	\$32.00	2.0%
2	BANK_B	\$70.00	0.0%
3	BANK_A	\$31.80	1.8%
3	BANK_B	\$73.00	3.0%
4	BANK_A	\$1.98	2.0%
4	BANK_B	\$25.00	15.0%
5	BANK_A	\$1.88	1.9%
5	BANK_B	\$25.00	15.0%
6	BANK_A	\$31.78	1.8%
6	BANK_B	\$70.00	0.0%
7	BANK_A	\$31.74	1.7%
7	BANK_B	\$70.00	0.0%
8	BANK_A	\$31.78	1.8%
8	BANK_B	\$70.00	10.0%
9	BANK_A	\$31.78	1.8%
9	BANK_B	\$70.00	0.0%
10	BANK_A	\$31.76	1.8%
10	BANK_B	\$70.00	0.0%
11	BANK_A	\$31.78	1.8%
11	BANK_B	\$70.00	0.0%

## C Experiment 2: Stochastic Environment - Detailed Results

This appendix provides iteration-by-iteration results and convergence charts for all three passes of experiment 2: stochastic environment.

### C.1 Pass 1

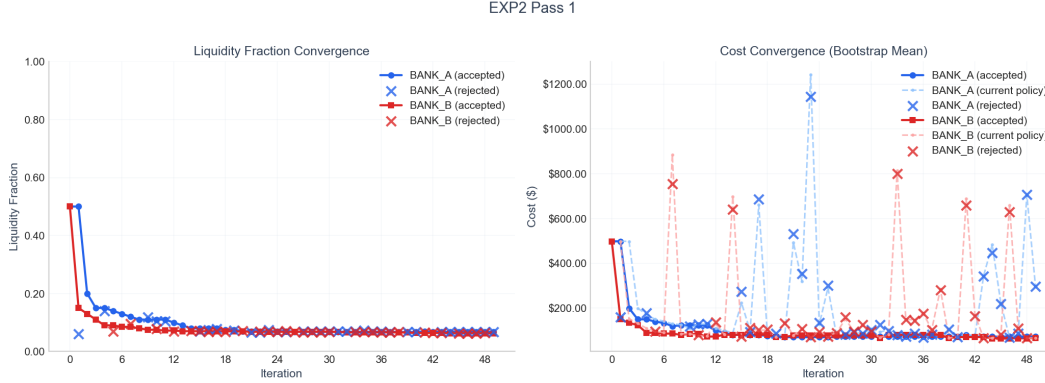


Figure 8: Experiment 2: Stochastic Environment - Pass 1 convergence

Table 13: Experiment 2: Stochastic Environment - Pass 1

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$498.00	50.0%
Baseline	BANK_B	\$498.00	50.0%
0	BANK_A	\$498.00	50.0%
0	BANK_B	\$498.00	50.0%
1	BANK_A	\$498.00	50.0%
1	BANK_B	\$150.43	15.0%
2	BANK_A	\$199.20	20.0%
2	BANK_B	\$131.29	13.0%
3	BANK_A	\$171.83	15.0%
3	BANK_B	\$109.56	11.0%
4	BANK_A	\$149.40	15.0%
4	BANK_B	\$95.73	9.0%
5	BANK_A	\$141.08	14.0%
5	BANK_B	\$89.95	9.0%
6	BANK_A	\$129.48	13.0%
6	BANK_B	\$884.73	8.5%
7	BANK_A	\$125.56	12.0%
7	BANK_B	\$84.72	8.5%
8	BANK_A	\$109.56	11.0%
8	BANK_B	\$85.07	8.0%
9	BANK_A	\$127.62	11.0%
9	BANK_B	\$76.03	7.5%

Continued on next page

Table 13 – continued from previous page

Iteration	Agent	Cost	Liquidity
10	BANK_A	\$128.05	11.0%
10	BANK_B	\$74.76	7.5%
11	BANK_A	\$109.56	11.0%
11	BANK_B	\$129.96	7.2%
12	BANK_A	\$99.60	10.0%
12	BANK_B	\$81.49	7.2%
13	BANK_A	\$89.64	9.0%
13	BANK_B	\$697.10	7.0%
14	BANK_A	\$222.85	8.0%
14	BANK_B	\$70.79	7.0%
15	BANK_A	\$98.01	8.0%
15	BANK_B	\$108.12	7.0%
16	BANK_A	\$660.68	8.0%
16	BANK_B	\$100.27	7.0%
17	BANK_A	\$79.68	8.0%
17	BANK_B	\$102.05	7.0%
18	BANK_A	\$85.02	7.5%
18	BANK_B	\$71.38	7.0%
19	BANK_A	\$74.76	7.5%
19	BANK_B	\$136.85	6.9%
20	BANK_A	\$493.15	7.0%
20	BANK_B	\$78.17	6.9%
21	BANK_A	\$318.12	7.0%
21	BANK_B	\$106.87	6.8%
22	BANK_A	\$1,242.11	7.0%
22	BANK_B	\$69.02	6.8%
23	BANK_A	\$127.76	7.0%
23	BANK_B	\$92.40	6.8%
24	BANK_A	\$290.25	7.0%
24	BANK_B	\$68.90	6.8%
25	BANK_A	\$74.18	7.0%
25	BANK_B	\$88.21	6.8%
26	BANK_A	\$83.10	6.8%
26	BANK_B	\$151.23	6.8%
27	BANK_A	\$89.60	6.8%
27	BANK_B	\$95.80	6.8%
28	BANK_A	\$87.05	6.8%
28	BANK_B	\$122.67	6.8%
29	BANK_A	\$106.49	6.8%
29	BANK_B	\$95.05	6.8%
30	BANK_A	\$92.87	6.8%
30	BANK_B	\$68.02	6.8%
31	BANK_A	\$101.46	6.8%
31	BANK_B	\$81.16	6.7%
32	BANK_A	\$77.13	6.8%

Continued on next page



Table 13 – continued from previous page

Iteration	Agent	Cost	Liquidity
32	BANK_B	\$812.94	6.6%
33	BANK_A	\$71.47	6.8%
33	BANK_B	\$149.65	6.6%
34	BANK_A	\$86.48	6.8%
34	BANK_B	\$146.90	6.6%
35	BANK_A	\$68.22	6.8%
35	BANK_B	\$177.67	6.6%
36	BANK_A	\$82.20	6.8%
36	BANK_B	\$103.09	6.6%
37	BANK_A	\$72.69	6.8%
37	BANK_B	\$283.89	6.6%
38	BANK_A	\$105.73	6.7%
38	BANK_B	\$66.33	6.6%
39	BANK_A	\$69.29	6.7%
39	BANK_B	\$70.93	6.5%
40	BANK_A	\$82.30	6.7%
40	BANK_B	\$688.53	6.4%
41	BANK_A	\$73.26	6.9%
41	BANK_B	\$169.62	6.4%
42	BANK_A	\$344.34	6.8%
42	BANK_B	\$65.13	6.4%
43	BANK_A	\$482.38	6.8%
43	BANK_B	\$65.67	6.4%
44	BANK_A	\$219.95	6.8%
44	BANK_B	\$81.49	6.3%
45	BANK_A	\$67.68	6.8%
45	BANK_B	\$659.08	6.3%
46	BANK_A	\$85.96	6.8%
46	BANK_B	\$108.63	6.3%
47	BANK_A	\$704.88	6.8%
47	BANK_B	\$65.53	6.3%
48	BANK_A	\$296.62	6.8%
48	BANK_B	\$66.37	6.3%
49	BANK_A	\$78.73	6.8%
49	BANK_B	\$196.58	6.2%

## C.2 Pass 2

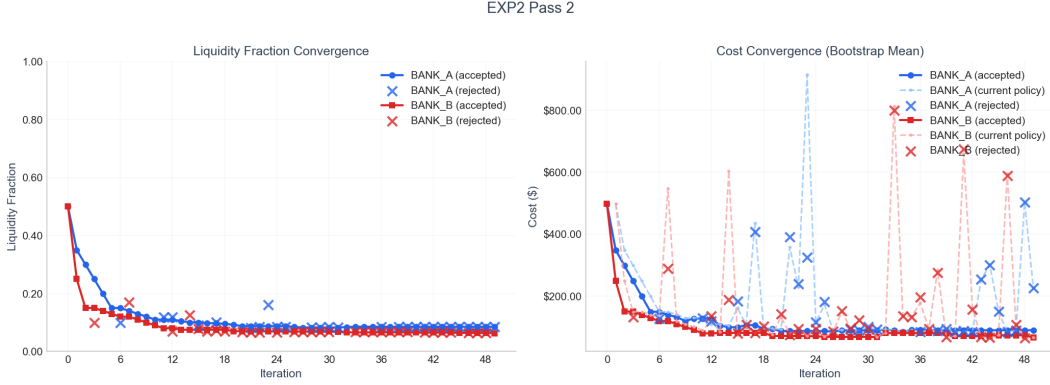


Figure 9: Experiment 2: Stochastic Environment - Pass 2 convergence

Table 14: Experiment 2: Stochastic Environment - Pass 2

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$498.00	50.0%
Baseline	BANK_B	\$498.00	50.0%
0	BANK_A	\$498.00	50.0%
0	BANK_B	\$498.00	50.0%
1	BANK_A	\$348.60	35.0%
1	BANK_B	\$249.00	25.0%
2	BANK_A	\$298.80	30.0%
2	BANK_B	\$149.40	15.0%
3	BANK_A	\$249.00	25.0%
3	BANK_B	\$149.40	15.0%
4	BANK_A	\$199.20	20.0%
4	BANK_B	\$139.44	14.0%
5	BANK_A	\$149.84	15.0%
5	BANK_B	\$129.48	13.0%
6	BANK_A	\$149.40	15.0%
6	BANK_B	\$548.36	12.0%
7	BANK_A	\$140.54	14.0%
7	BANK_B	\$119.52	12.0%
8	BANK_A	\$129.48	13.0%
8	BANK_B	\$109.56	11.0%
9	BANK_A	\$129.99	12.0%
9	BANK_B	\$100.61	10.0%
10	BANK_A	\$128.05	11.0%
10	BANK_B	\$89.64	9.0%
11	BANK_A	\$109.56	11.0%
11	BANK_B	\$120.59	8.0%
12	BANK_A	\$109.56	11.0%

Continued on next page

Table 14 – continued from previous page

Iteration	Agent	Cost	Liquidity
12	BANK_B	\$84.61	8.0%
13	BANK_A	\$104.64	10.5%
13	BANK_B	\$603.95	7.5%
14	BANK_A	\$161.57	10.0%
14	BANK_B	\$74.76	7.5%
15	BANK_A	\$107.39	10.0%
15	BANK_B	\$101.82	7.5%
16	BANK_A	\$435.60	9.8%
16	BANK_B	\$80.46	7.5%
17	BANK_A	\$97.08	9.8%
17	BANK_B	\$99.14	7.5%
18	BANK_A	\$95.61	9.5%
18	BANK_B	\$76.15	7.5%
19	BANK_A	\$92.16	9.2%
19	BANK_B	\$132.77	7.0%
20	BANK_A	\$358.12	8.8%
20	BANK_B	\$77.86	7.0%
21	BANK_A	\$229.23	8.8%
21	BANK_B	\$97.13	7.0%
22	BANK_A	\$914.93	8.8%
22	BANK_B	\$71.15	7.0%
23	BANK_A	\$114.79	8.8%
23	BANK_B	\$91.48	6.9%
24	BANK_A	\$173.05	8.8%
24	BANK_B	\$68.76	6.9%
25	BANK_A	\$87.72	8.8%
25	BANK_B	\$86.01	6.9%
26	BANK_A	\$91.77	8.5%
26	BANK_B	\$150.25	6.9%
27	BANK_A	\$91.88	8.2%
27	BANK_B	\$94.50	6.9%
28	BANK_A	\$88.68	8.2%
28	BANK_B	\$121.58	6.9%
29	BANK_A	\$99.88	8.2%
29	BANK_B	\$95.22	6.9%
30	BANK_A	\$92.82	8.2%
30	BANK_B	\$68.62	6.9%
31	BANK_A	\$92.70	8.2%
31	BANK_B	\$81.12	6.8%
32	BANK_A	\$88.92	8.4%
32	BANK_B	\$812.94	6.6%
33	BANK_A	\$85.20	8.6%
33	BANK_B	\$135.58	6.6%
34	BANK_A	\$90.94	8.5%
34	BANK_B	\$133.66	6.6%

Continued on next page

Table 14 – continued from previous page

Iteration	Agent	Cost	Liquidity
35	BANK_A	\$84.84	8.5%
35	BANK_B	\$177.67	6.6%
36	BANK_A	\$89.92	8.5%
36	BANK_B	\$94.99	6.6%
37	BANK_A	\$87.83	8.5%
37	BANK_B	\$283.89	6.6%
38	BANK_A	\$94.91	8.5%
38	BANK_B	\$66.33	6.6%
39	BANK_A	\$84.72	8.5%
39	BANK_B	\$71.89	6.6%
40	BANK_A	\$87.04	8.5%
40	BANK_B	\$659.82	6.5%
41	BANK_A	\$85.21	8.5%
41	BANK_B	\$159.79	6.5%
42	BANK_A	\$254.35	8.5%
42	BANK_B	\$67.79	6.5%
43	BANK_A	\$300.64	8.5%
43	BANK_B	\$65.23	6.5%
44	BANK_A	\$149.41	8.5%
44	BANK_B	\$74.53	6.5%
45	BANK_A	\$84.72	8.5%
45	BANK_B	\$595.24	6.4%
46	BANK_A	\$89.33	8.5%
46	BANK_B	\$107.76	6.4%
47	BANK_A	\$503.13	8.5%
47	BANK_B	\$63.72	6.4%
48	BANK_A	\$226.15	8.5%
48	BANK_B	\$67.05	6.4%
49	BANK_A	\$88.50	8.5%
49	BANK_B	\$148.62	6.3%

### C.3 Pass 3

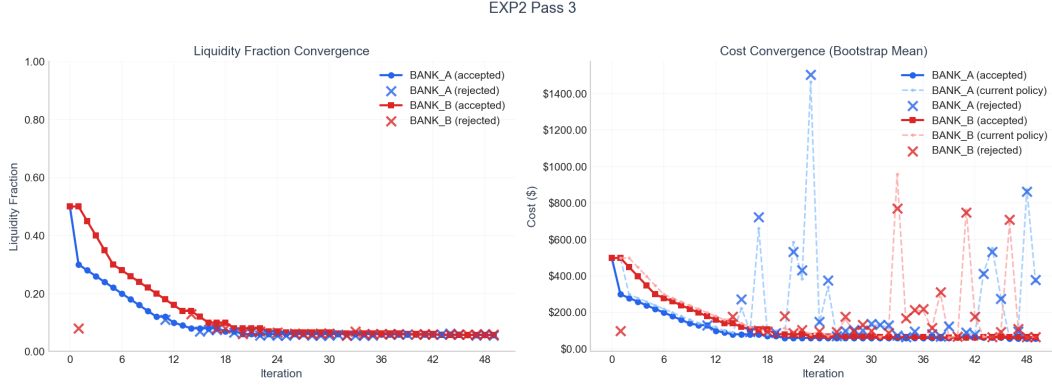


Figure 10: Experiment 2: Stochastic Environment - Pass 3 convergence

Table 15: Experiment 2: Stochastic Environment - Pass 3

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$498.00	50.0%
Baseline	BANK_B	\$498.00	50.0%
0	BANK_A	\$498.00	50.0%
0	BANK_B	\$498.00	50.0%
1	BANK_A	\$298.80	30.0%
1	BANK_B	\$498.00	50.0%
2	BANK_A	\$278.88	28.0%
2	BANK_B	\$448.20	45.0%
3	BANK_A	\$258.96	26.0%
3	BANK_B	\$398.40	40.0%
4	BANK_A	\$239.04	24.0%
4	BANK_B	\$348.60	35.0%
5	BANK_A	\$219.12	22.0%
5	BANK_B	\$298.80	30.0%
6	BANK_A	\$199.20	20.0%
6	BANK_B	\$278.88	28.0%
7	BANK_A	\$179.28	18.0%
7	BANK_B	\$258.96	26.0%
8	BANK_A	\$159.36	16.0%
8	BANK_B	\$239.04	24.0%
9	BANK_A	\$141.88	14.0%
9	BANK_B	\$219.12	22.0%
10	BANK_A	\$129.24	12.0%
10	BANK_B	\$199.20	20.0%
11	BANK_A	\$119.52	12.0%
11	BANK_B	\$179.28	18.0%
12	BANK_A	\$99.60	10.0%

Continued on next page

Table 15 – continued from previous page

Iteration	Agent	Cost	Liquidity
12	BANK_B	\$159.36	16.0%
13	BANK_A	\$89.64	9.0%
13	BANK_B	\$166.79	14.0%
14	BANK_A	\$222.85	8.0%
14	BANK_B	\$139.44	14.0%
15	BANK_A	\$98.01	8.0%
15	BANK_B	\$119.76	12.0%
16	BANK_A	\$660.68	8.0%
16	BANK_B	\$105.57	10.0%
17	BANK_A	\$79.68	8.0%
17	BANK_B	\$102.06	10.0%
18	BANK_A	\$84.60	7.0%
18	BANK_B	\$99.60	10.0%
19	BANK_A	\$69.72	7.0%
19	BANK_B	\$114.10	8.0%
20	BANK_A	\$584.18	6.0%
20	BANK_B	\$89.68	8.0%
21	BANK_A	\$384.79	6.0%
21	BANK_B	\$103.56	8.0%
22	BANK_A	\$1,464.39	6.0%
22	BANK_B	\$79.68	8.0%
23	BANK_A	\$142.58	6.0%
23	BANK_B	\$94.76	7.0%
24	BANK_A	\$338.92	6.0%
24	BANK_B	\$70.94	7.0%
25	BANK_A	\$70.16	6.0%
25	BANK_B	\$92.40	6.5%
26	BANK_A	\$97.81	6.0%
26	BANK_B	\$162.47	6.5%
27	BANK_A	\$97.82	6.0%
27	BANK_B	\$95.79	6.5%
28	BANK_A	\$86.73	6.0%
28	BANK_B	\$127.63	6.5%
29	BANK_A	\$128.02	6.0%
29	BANK_B	\$95.47	6.5%
30	BANK_A	\$131.51	6.0%
30	BANK_B	\$65.14	6.5%
31	BANK_A	\$116.39	6.0%
31	BANK_B	\$83.30	6.2%
32	BANK_A	\$75.69	6.0%
32	BANK_B	\$958.00	6.2%
33	BANK_A	\$67.55	6.0%
33	BANK_B	\$160.58	6.2%
34	BANK_A	\$90.06	6.0%
34	BANK_B	\$210.88	6.2%

Continued on next page

Table 15 – continued from previous page

Iteration	Agent	Cost	Liquidity
35	BANK_A	\$60.30	6.0%
35	BANK_B	\$209.37	6.2%
36	BANK_A	\$80.84	5.9%
36	BANK_B	\$112.61	6.2%
37	BANK_A	\$65.63	5.9%
37	BANK_B	\$305.89	6.2%
38	BANK_A	\$121.10	5.9%
38	BANK_B	\$63.02	6.2%
39	BANK_A	\$63.52	5.9%
39	BANK_B	\$68.72	6.0%
40	BANK_A	\$90.92	5.8%
40	BANK_B	\$762.87	6.0%
41	BANK_A	\$76.82	5.8%
41	BANK_B	\$173.25	6.0%
42	BANK_A	\$419.31	5.8%
42	BANK_B	\$66.46	6.0%
43	BANK_A	\$550.59	5.8%
43	BANK_B	\$64.63	5.8%
44	BANK_A	\$283.08	5.8%
44	BANK_B	\$94.72	5.8%
45	BANK_A	\$58.74	5.8%
45	BANK_B	\$691.13	5.8%
46	BANK_A	\$93.98	5.7%
46	BANK_B	\$113.72	5.8%
47	BANK_A	\$874.88	5.7%
47	BANK_B	\$62.82	5.8%
48	BANK_A	\$370.30	5.7%
48	BANK_B	\$65.30	5.8%
49	BANK_A	\$70.88	5.7%
49	BANK_B	\$309.02	5.8%

## D Experiment 3: Symmetric Scenario - Detailed Results

This appendix provides iteration-by-iteration results and convergence charts for all three passes of experiment 3: symmetric scenario.

## D.1 Pass 1



Figure 11: Experiment 3: Symmetric Scenario - Pass 1 convergence

Table 16: Experiment 3: Symmetric Scenario - Pass 1

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$49.95	50.0%
Baseline	BANK_B	\$49.95	50.0%
0	BANK_A	\$49.95	50.0%
0	BANK_B	\$49.95	50.0%
1	BANK_A	\$29.97	30.0%
1	BANK_B	\$39.96	40.0%
2	BANK_A	\$120.99	1.0%
2	BANK_B	\$69.97	30.0%
3	BANK_A	\$120.90	0.9%
3	BANK_B	\$68.98	29.0%
4	BANK_A	\$120.96	1.0%
4	BANK_B	\$69.97	30.0%
5	BANK_A	\$120.99	1.0%
5	BANK_B	\$71.98	32.0%
6	BANK_A	\$120.96	1.0%
6	BANK_B	\$69.97	30.0%
7	BANK_A	\$120.99	1.0%
7	BANK_B	\$69.97	30.0%



## D.2 Pass 2

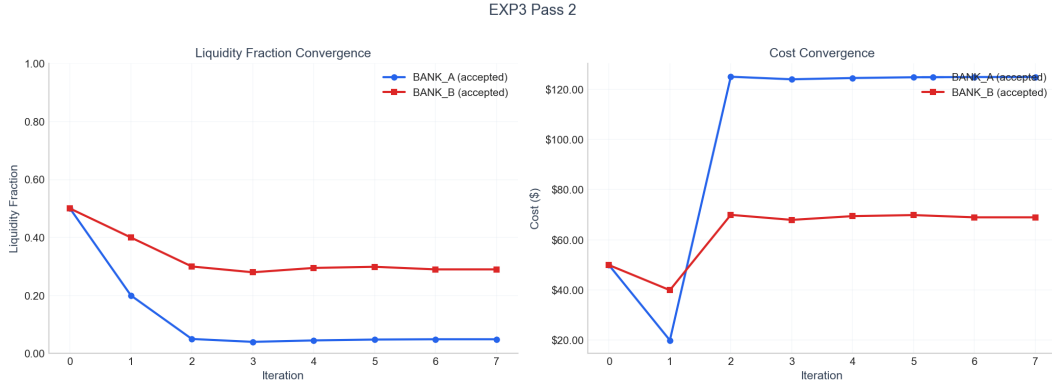


Figure 12: Experiment 3: Symmetric Scenario - Pass 2 convergence

Table 17: Experiment 3: Symmetric Scenario - Pass 2

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$49.95	50.0%
Baseline	BANK_B	\$49.95	50.0%
0	BANK_A	\$49.95	50.0%
0	BANK_B	\$49.95	50.0%
1	BANK_A	\$19.98	20.0%
1	BANK_B	\$39.96	40.0%
2	BANK_A	\$125.01	5.0%
2	BANK_B	\$69.97	30.0%
3	BANK_A	\$123.99	4.0%
3	BANK_B	\$67.96	28.0%
4	BANK_A	\$124.50	4.5%
4	BANK_B	\$69.46	29.5%
5	BANK_A	\$124.80	4.8%
5	BANK_B	\$69.88	29.9%
6	BANK_A	\$124.89	4.9%
6	BANK_B	\$68.98	29.0%
7	BANK_A	\$124.89	4.9%
7	BANK_B	\$68.98	29.0%

### D.3 Pass 3



Figure 13: Experiment 3: Symmetric Scenario - Pass 3 convergence

Table 18: Experiment 3: Symmetric Scenario - Pass 3

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	\$49.95	50.0%
Baseline	BANK_B	\$49.95	50.0%
0	BANK_A	\$49.95	50.0%
0	BANK_B	\$49.95	50.0%
1	BANK_A	\$19.98	20.0%
1	BANK_B	\$19.98	20.0%
2	BANK_A	\$209.99	10.0%
2	BANK_B	\$200.99	1.0%
3	BANK_A	\$209.00	9.0%
3	BANK_B	\$200.51	0.5%
4	BANK_A	\$209.99	10.0%
4	BANK_B	\$200.90	0.9%
5	BANK_A	\$207.98	8.0%
5	BANK_B	\$200.99	1.0%
6	BANK_A	\$209.90	9.9%
6	BANK_B	\$200.96	0.9%
7	BANK_A	\$209.96	10.0%
7	BANK_B	\$200.96	0.9%

## E LLM System Prompt Documentation

This appendix documents the system prompt provided to LLM agents during policy optimization. The content is extracted programmatically from the SimCash codebase to ensure this documentation remains synchronized with the actual implementation.

The system prompt establishes the agent’s role, provides domain context, and specifies the format requirements for policy proposals. Additional sections (policy schema, cost rates, and constraint-specific guidance) are injected dynamically based on experiment configuration.

## E.1 Expert Introduction

The prompt begins by establishing the agent's role as a payment system optimization expert:

You are an expert in payment system optimization.

Your job is to generate valid JSON policies for the SimCash payment simulator.

You are an optimization agent in a simulation of an interbank payment settlement in a real-time gross settlement (RTGS) environment. Each agent represents a bank with a settlement account at the central bank.

## E.2 Domain Context

The agent receives detailed context about RTGS settlement mechanics:

## Domain Context: Interbank Payment Settlement

### Real-Time Gross Settlement (RTGS)

Payments arrive throughout the simulated trading day with specified amounts, counterparties, deadlines, and priority levels. Settlement occurs immediately when the sending bank has sufficient balance or available credit.

### Queuing Mechanism

When liquidity is insufficient, payments enter a queue:

- **Queue 1**: Immediate settlement attempts (holds payments briefly)
- **Queue 2**: Longer-term holding when liquidity is constrained

Queued payments accumulate delay costs until settled.

### Key Concepts

- **Balance**: Current reserves in settlement account (integer cents)
- **Effective Liquidity**: Balance + credit limit - pending obligations
- **Credit Limit**: Available daylight overdraft or collateralized credit
- **Collateral**: Assets posted to central bank to secure credit

## E.3 Cost Structure and Objectives

The optimization objective and cost components are explained:

## Cost Structure and Objectives

**Your objective is to minimize total cost.**

Costs include:

1. **Overdraft Charges**: Basis points on negative balance positions
2. **Delay Penalties**: Per-tick costs for each transaction waiting in queue
3. **Deadline Penalties**: One-time charge when a payment becomes overdue
4. **Overdue Multiplier**: Increased delay costs after deadline passes
5. **End-of-Day Penalties**: Severe charges for unsettled payments at close

### ### Why This is Non-Trivial

Actions have delayed consequences:

- Releasing liquidity early may reduce delay cost but increase overdraft exposure
- Holding payments preserves liquidity but risks deadline penalties
- Optimal behavior requires balancing immediate costs against future states

### ### Strategic Considerations

- High-priority payments have higher delay costs
- Payments close to deadline should often be released
- Incoming payments may provide liquidity to release queued payments
- End-of-day penalties are typically very high - avoid unsettled transactions

## E.4 Optimization Process

The iterative optimization workflow is described:

### ## Optimization Process

You will be provided with:

1. **\*\*Current policy tree\*\***: The policy to improve
2. **\*\*Simulation output\*\***: Tick-by-tick logs from recent runs
3. **\*\*Iteration history\*\***: How the policy has evolved and cost changes

### ### Your Task

Analyze the provided data, identify inefficiencies or suboptimal decisions, and propose modifications to the policy tree that reduce total costs.

Focus on:

- Decisions that led to deadline penalties or high delay costs
- Opportunities to release payments earlier with available liquidity
- Conditions that are too aggressive (causing overdrafts) or too conservative

### ### Output Requirements

Return a complete, valid JSON policy with:

- All tree types that are enabled in this scenario
- All parameters defined before they are referenced
- Unique node\_id for every node

## E.5 Pre-Generation Checklist

Before generating policies, agents must verify compliance:

```
#####  
#                                MANDATORY PRE-GENERATION CHECKLIST                                #  
#####
```

BEFORE generating ANY policy, verify you will satisfy ALL of these:

- [ ] Every {"param": "X"} has a matching "X" key in the "parameters" object

- [ ] Every action matches its tree type (see allowed actions below)
- [ ] Every node has a unique "node\_id" string
- [ ] Arithmetic expressions are wrapped in {"compute": {...}}
- [ ] Only use fields and parameters from the ALLOWED sections
- [ ] No undefined field references
- [ ] No mixing of action types between trees

#####

## E.6 Final Instructions

The prompt concludes with output requirements:

```
#####
#                               FINAL INSTRUCTIONS                               #
#####
```

1. Generate a COMPLETE policy JSON with all required trees
2. Ensure EVERY node has a unique node\_id
3. Define ALL parameters before referencing them
4. Use ONLY allowed actions for each tree type
5. Wrap ALL arithmetic in {"compute": {...}}
6. Keep trees reasonably simple (3-5 levels max) for robustness
7. Focus improvements on areas identified in the simulation output

The policy MUST be syntactically valid JSON that passes validation.

## E.7 Dynamic Sections

The following sections are injected dynamically based on experiment configuration. We show examples from Experiment 2 (12-Period Stochastic) to illustrate the content.

### E.7.1 Experiment Customization (Exp 2)

Each experiment can provide scenario-specific guidance:

```
#####
#                               EXPERIMENT CUSTOMIZATION                               #
#####
```

This scenario tests a fundamental tradeoff in payment systems:

- Allocating liquidity from the pool allows you to settle payments
- But allocated liquidity has an opportunity cost

The KEY DECISION in this scenario is: What fraction of the liquidity pool should be allocated at the START of the day?

IMPORTANT: Focus your optimization on the 'initial\_liquidity\_fraction' parameter. The payment\_tree should remain a simple Release action - the real optimization

is in how much liquidity to commit upfront.

With a hard liquidity constraint (no overdraft allowed), you must have sufficient balance to settle each payment. Incoming payments from the counterparty provide liquidity that can be recycled for your outgoing payments.

#####

## E.7.2 Policy Constraints (Exp 2)

The policy schema is filtered to show only allowed elements:

### ### ALLOWED PARAMETERS

Define these in the 'parameters' object:

```
initial_liquidity_fraction (float): 0.0 to 1.0
    Fraction of liquidity_pool to allocate at simulation start. Value between 0.0 and 1.0.
```

### ### ALLOWED FIELDS

Reference with {"field": "name"}:

- system\_tick\_in\_day
- balance
- amount
- remaining\_amount
- ticks\_to\_deadline

### ### ALLOWED ACTIONS BY TREE TYPE

```
payment_tree: Release, Hold
bank_tree: NoAction
```

## E.7.3 Cost Parameters (Exp 2)

Current cost rates from the experiment configuration:

### ## COST PARAMETERS

Per-Tick Costs:

- liquidity\_cost\_per\_tick\_bps: 83 (0.1 / 12 ticks)
- delay\_cost\_per\_tick\_per\_cent: 0.2

One-Time Costs:

- deadline\_penalty: 50,000 cents
- eod\_penalty\_per\_transaction: 100,000 cents

Disabled in this scenario:

- overdraft\_bps\_per\_tick: 0 (hard liquidity constraint)
- collateral\_cost\_per\_tick\_bps: 0 (using liquidity\_pool mode)

## E.8 User Prompt Example (Exp 2, Pass 2, Iteration 6)

Each iteration, agents receive a comprehensive user prompt containing their performance history and current context. The full prompt (~10,000 tokens) includes the following sections. We show condensed excerpts from BANK\_A’s prompt at iteration 6 of Experiment 2, Pass 2:

### E.8.1 Current State Summary

#### POLICY OPTIMIZATION CONTEXT - BANK\_A - ITERATION 6

This document provides complete context for optimizing YOUR payment policy. Analyze the simulation outputs and historical data to identify improvements.

NOTE: You are optimizing policy for BANK\_A ONLY. Focus on YOUR decisions.

#### TABLE OF CONTENTS:

1. Current State Summary
2. Cost Analysis
3. Optimization Guidance
4. Simulation Output
5. Full Iteration History
6. Parameter Trajectories
7. Final Instructions

#### ## 1. CURRENT STATE SUMMARY

##### ### Performance Metrics (Iteration 6)

Metric	Value
**Mean Total Cost**	\$14,984 (0.3% from previous)
**Cost Std Dev**	\$1,046
**Sample Cost**	\$12,948 (Seed #1547979735)
**Settlement Rate**	0.0%
**Failure Rate**	0%

##### ### Current Policy Parameters (BANK\_A)

```

'''json
{
  "initial_liquidity_fraction": 0.15
}
'''

```

## E.8.2 Cost Breakdown and Rates

### ## 2. COST ANALYSIS

#### ### Cost Breakdown (Last Iteration)

Cost Type	Amount	% of Total	Priority
delay_cost	\$22	100.0%	HIGH
overdraft_cost	\$0	0.0%	LOW
deadline_penalty	\$0	0.0%	LOW
eod_penalty	\$0	0.0%	LOW

#### ### Cost Rate Configuration

```
'''json
{
  "overdraft_bps_per_tick": 0.0,
  "delay_cost_per_tick_per_cent": 0.2,
  "collateral_cost_per_tick_bps": 0.0,
  "eod_penalty_per_transaction": 100000,
  "deadline_penalty": 50000,
  "split_friction_cost": 0,
  "overdue_delay_multiplier": 5.0,
  "liquidity_cost_per_tick_bps": 83.0
}
'''
```

## E.8.3 Iteration History and Parameter Trajectories

### ## 5. FULL ITERATION HISTORY

#### ### Metrics Summary Table

Iter	Status	Mean Cost	Std Dev	Settlement	Best Seed	Worst Seed
1	BEST	\$34,860	±\$0	0.0%	\$0	\$0
2	BEST	\$29,880	±\$0	0.0%	\$0	\$0
3	BEST	\$24,900	±\$0	0.0%	\$0	\$0
4	BEST	\$19,920	±\$0	0.0%	\$0	\$0
5	BEST	\$14,940	±\$0	0.0%	\$0	\$0

#### ### Current Best Policy

The best policy so far was discovered in **iteration 5** with mean cost **\$14,940**.

#### ### Detailed Changes Per Iteration

##### #### Iteration 1 (BEST POLICY)



```

**Performance:** Mean cost $34,860, Settlement 0.0%

**BANK_A Parameters at this iteration:**
'''json
{
  "initial_liquidity_fraction": 0.35
}
'''

#### Iteration 2 (BEST POLICY)

**Performance:** Mean cost $29,880, Settlement 0.0%
**Comparison:** -$49.80 vs best (NEW BEST)

**BANK_A Changes:**
- Changed 'initial_liquidity_fraction': 0.35 0.3 (0.05)

**BANK_A Parameters at this iteration:**
'''json
{
  "initial_liquidity_fraction": 0.3
}
'''

#### Iteration 3 (BEST POLICY)

**Performance:** Mean cost $24,900, Settlement 0.0%
**Comparison:** -$49.80 vs best (NEW BEST)

**BANK_A Changes:**
- Changed 'initial_liquidity_fraction': 0.3 0.25 (0.05)

**BANK_A Parameters at this iteration:**
'''json
{
  "initial_liquidity_fraction": 0.25
}
'''

#### Iteration 4 (BEST POLICY)

**Performance:** Mean cost $19,920, Settlement 0.0%
**Comparison:** -$49.80 vs best (NEW BEST)

**BANK_A Changes:**
- Changed 'initial_liquidity_fraction': 0.25 0.2 (0.05)

```

```

**BANK_A Parameters at this iteration:**
'''json
{
  "initial_liquidity_fraction": 0.2
}
'''

#### Iteration 5 (BEST POLICY)

**Performance:** Mean cost $14,940, Settlement 0.0%
**Comparison:** -$49.80 vs best (NEW BEST)

**BANK_A Changes:**
  - Changed 'initial_liquidity_fraction': 0.2  0.15 (0.05)

**BANK_A Parameters at this iteration:**
'''json
{
  "initial_liquidity_fraction": 0.15
}
'''

```

## ## 6. PARAMETER TRAJECTORIES

Track how each BANK\_A parameter evolved across iterations:

### initial\_liquidity\_fraction

Iteration	Value
1	0.350
2	0.300
3	0.250
4	0.200
5	0.150

\*Overall: decreased 57.1% from 0.350 to 0.150\*

## E.8.4 Final Instructions

### ## 7. FINAL INSTRUCTIONS

Based on the above analysis, generate an improved policy for **\*\*BANK\_A\*\*** that:

1. **\*\*Beats the current best policy\*\*** - your policy must have LOWER cost than the best
2. **\*\*Maintains 100% settlement rate\*\*** - this is non-negotiable
3. **\*\*Makes incremental adjustments\*\*** - avoid drastic changes unless clearly needed

4. **\*\*Learns from REJECTED policies\*\*** - don't repeat changes that made things worse

**\*\*Current Best\*\***: Iteration 5 with mean cost \$14,940.

Your goal is to beat this. If your policy is worse, it will be rejected and we will continue optimizing from the current best policy.

**### What to Consider:**

- **\*\*Simulation trace analysis\*\***: What decisions in seed #1547979735 drove costs?
- **\*\*Cost breakdown\*\***: Which cost types (delay, collateral, overdraft) dominate?
- **\*\*REJECTED policies\*\***: Why did they fail? What changes should you avoid?
- **\*\*Parameter trends\*\***: Which parameters correlate with cost improvements?
- **\*\*Trade-offs\*\***: Balance delay costs vs collateral costs vs overdraft costs

**### Output Requirements:**

Generate a complete, valid policy JSON that:

- Defines all parameters before using them
- Uses only allowed fields and actions
- Includes unique node\_id for every node
- Wraps arithmetic in {"compute": {...}}

Focus your changes on the areas with highest impact potential. Remember: if your policy is worse than the current best, it will be REJECTED and you'll need to try a different approach.

=====

CURRENT POLICY FOR BANK\_A

=====

```
'''json
{
  "version": "2.0",
  "policy_id": "BANK_A_liquidity_opt_iter6",
  "parameters": {
    "initial_liquidity_fraction": 0.15
  },
  "bank_tree": {
    "type": "action",
    "node_id": "B0_no_action_iter6",
    "action": "NoAction"
  },
  "payment_tree": {
    "type": "action",
    "node_id": "P0_release_all_iter6",
    "action": "Release"
  }
}
```

'''

Current policy:

```
{
  "version": "2.0",
  "policy_id": "BANK_A_liquidity_opt_iter6",
  "parameters": {
    "initial_liquidity_fraction": 0.15
  },
  "bank_tree": {
    "type": "action",
    "node_id": "B0_no_action_iter6",
    "action": "NoAction"
  },
  "payment_tree": {
    "type": "action",
    "node_id": "P0_release_all_iter6",
    "action": "Release"
  }
}
```

Generate an improved policy that reduces total cost.  
Output ONLY the JSON policy, no explanation.

### E.8.5 Simulation Trace (Exp 2, Pass 2)

The prompt includes a tick-by-tick simulation trace showing transaction arrivals, settlements, and balance changes. This example is from Experiment 2 (stochastic scenario) showing the first 40 events from a representative simulation:

```
Tick 0
Deferred Credit: BANK_A received $301.58
RtgsSubmission: fc963c63
RTGS Settled: BANK_B BANK_A | $186.57
  Balance: $534.99 $348.42
Costs (BANK_A): total: $7.06
Submit: TX 0253cb8a...
Arrival: BANK_B BANK_A | $45.36 | Deadline: Tick 4
RTGS Settled: BANK_B BANK_A | $45.36
  Balance: $650.00 $604.64
RTGS Settled: BANK_A BANK_B | $92.78
  Balance: $851.00 $758.22
Costs (BANK_B): total: $5.40
Submit: TX 3757d3cd...
Deferred Credit: BANK_B received $92.78
RtgsSubmission: 3757d3cd
RtgsSubmission: 0253cb8a
Submit: TX 46436b60...
```

Arrival: BANK\_A BANK\_B | \$92.78 | Deadline: Tick 8  
RTGS Settled: BANK\_B BANK\_A | \$69.65  
Balance: \$604.64 \$534.99  
Arrival: BANK\_B BANK\_A | \$186.57 | Deadline: Tick 7  
RtgsSubmission: 46436b60  
Arrival: BANK\_B BANK\_A | \$69.65 | Deadline: Tick 6  
Submit: TX fc963c63...

Tick 1

Costs (BANK\_B): total: \$5.40  
Deferred Credit: BANK\_B received \$111.10  
RTGS Settled: BANK\_A BANK\_B | \$57.43  
Balance: \$1,006.13 \$948.70  
Submit: TX e94b6a42...  
Arrival: BANK\_A BANK\_B | \$53.67 | Deadline: Tick 8  
Arrival: BANK\_B BANK\_A | \$77.62 | Deadline: Tick 8  
RtgsSubmission: e94b6a42  
Submit: TX f18e9699...  
RTGS Settled: BANK\_B BANK\_A | \$51.07  
Balance: \$441.20 \$390.13  
RtgsSubmission: f18e9699  
Deferred Credit: BANK\_A received \$272.98  
Submit: TX 9dbe9b13...  
Arrival: BANK\_B BANK\_A | \$144.29 | Deadline: Tick 9  
Submit: TX da729533...  
Arrival: BANK\_A BANK\_B | \$57.43 | Deadline: Tick 5  
RtgsSubmission: f8ed1e01  
RtgsSubmission: da729533  
RtgsSubmission: 9dbe9b13  
Costs (BANK\_A): total: \$7.06  
RTGS Settled: BANK\_A BANK\_B | \$53.67  
Balance: \$1,059.80 \$1,006.13