GAME-THEORETIC SEARCH IN PHYSICAL SYSTEMS WITH AN APPLICATION TO THE SPORT OF CURLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Game-theoretic reinforcement-learning systems have achieved strong results in board and card games, but extending them to physical sports requires reasoning over continuous actions, stochastic execution, and nontrivial physics. We introduce a search-and-distillation framework for the sport of mixed doubles curling, using a physics simulator with iterative empirical-game optimization. A graph transformer jointly represents shot selection and position value: a Gaussian-mixture policy models distinct continuous shot modes, a distributional value head predicts end-score outcomes, and an auxiliary latent-dynamics objective encourages world-model-like physics understanding. Starting from human match data, we iteratively compute a meta-Nash mixture over existing policies, search for an approximate best response against that mixture, and distill accepted search decisions and realized returns into the next policy. Under human-like imperfect execution noise, the final policy beats open-weight language-model agents by $+0.96$ points per end. Against frontier agentic models like Claude Opus 5 and GPT-6-Astra, which are becoming powerful across diverse games, our policy wins by at least $+0.19$ points per end. In blind paired comparisons on 30 held-out positions, two U.S. national (Olympic) curling team staff members judged the policy's shots comparably to recorded human actions, selecting the policy as more optimal in 26 of 60 judgments (43%). These results demonstrate a practical framework for combining empirical-game reasoning, continuous-action search, and physical simulation in stochastic sports.

1 INTRODUCTION

Game-theoretic reinforcement learning has reached superhuman performance in static board games like Go, chess, shogi, and poker (Silver et al., 2018; Moravčík et al., 2017), with newer improvements in more continuous games (Lee et al., 2018; Han et al., 2022). Mostly, such gameplay consists of decisions that can be enumerated or made tractable through game-tree decomposition. Curling retains alternating, zero-sum play but each move is a *continuous* physical control whose noisy trajectory and collisions determine the next state (Yee et al., 2016; Ahmad et al., 2016).

In curling, two teams take turns sliding granite stones down an ice sheet toward concentric rings called the house. Each stone remains in play unless knocked out or thrown out of bounds. Games are played in ends (similar to innings), where the team with the stone closest to the center ring (the button) scores one point for each stone closer than the opponent's best. If no stones rest in the house, the end is blanked. The team throwing last holds the hammer, which is a key strategic advantage that passes to whoever failed to score in the prior end. A stone placed to protect another is a guard.

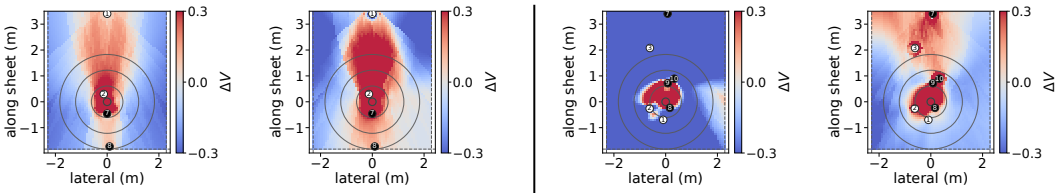
We focus on mixed doubles, a fast-paced two-player Olympic variant played over eight ends. Teams throw five stones per end around two pre-placed stones: a center guard for the first-throwing team and a house stone for the team with hammer. Once per game, the team with hammer can use a power play to shift these pre-placed stones laterally to clear the center. This compact format features just ten throws per end, short games, and high-stakes shots.

Our goal is both strategic and explanatory: approach an optimal strategy in our game model while producing recommendations and position evaluations that curlers can inspect, abstracting away brushing to focus on the release. This leads to four design requirements:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1. a multimodal policy coupled to explicit search;
2. a value distribution, not only a point estimate;
3. a realistic, tunable simulator for planning and counterfactual analysis (Xiao et al., 2023);
4. an action-conditioned latent model that improves the shared representation without replacing the simulator in search (Schrittwieser et al., 2020; Ye et al., 2021; Hafner et al., 2023).

We encode stones, landmarks, and game context as a graph. A shared graph transformer produces a Gaussian-mixture policy over release parameters, a distribution over end score, and an action-conditioned next-state latent. The latent head learns from simulator transitions (Grill et al., 2020); planning itself uses the explicit physics simulator. Figure 1 compares the learned value surface with its human-initialized prior.



(a) Human Prior (b) Ours (c) Human Prior (d) Ours

Figure 1: Value visualization for two example boards, comparing the inverse-recovered human prior with our final policy π_3. Each pixel shows predicted value if the thrown stone stopped there without collision; red is better for the player to move. The colour limit is fixed at ± 0.3 across all four panels, so the hot core is saturated. We observe nontrivial zones of high value (denoted in red), like those around blockers near the button center.

We initialize from human matches, then run three empirical-game response rounds. Each round solves a meta-Nash mixture over the current population, collects a searched approximate best response, and distills accepted actions and realized outcomes into a new policy. Three rounds do not establish equilibrium in the continuous game; we stop early because each approximate best response requires expensive simulator search. Our goal is to test whether a small number of population updates yields measurable strategic improvement, rather than to claim convergence of the continuous game. We expect large improvement with further iteration.

The final policy has a positive score differential against the previous meta-Nash and both earlier responses. It beats hosted open-weight language model agents. Of two frontier agents, it is ahead of GPT-6-Astra on standard ends, though not significantly, and level there with the stronger, Claude Opus 5, which it then outscores on balanced power-play ends. In a blind comparison on 30 recorded positions, U.S. Olympic Curling Team staff showed no significant preference between its shots and human shots. Our contribution is the joint use of graph-based policy and value learning, simulator-grounded search under execution noise, latent physics supervision, and empirical-game response training.

Our technical contributions are a graph-transformer policy and probabilistic value model for continuous shot selection, simulator-grounded KR-UCT Search (Yee et al., 2016) under execution noise, and an iterative empirical-game best-response procedure that distills searched actions while using latent-dynamics consistency to shape the shared representation.

2 Related Work

We draw on four lines of work: strategic decision-making in curling, robotic curling, physics-based strategy in other sports, and world models for reinforcement learning.

Strategy and decision-making in curling. A growing body of work studies strategic decision-making in curling using machine learning and search. Xiao et al. (2023) propose a policy decision system designed for real competition scenes, combining a situation-aware perception network with a digital extraction module and a domain-specific Curling MCTS that plans in a continuous action

space. Brunner & Riesen (2024) compare supervised learning methods for position analysis in simulated curling and show that neural and graph-based models outperform heuristic baselines, while emphasizing that existing datasets remain insufficient to consistently surpass professional curlers, particularly in complex mid-end situations. Han et al. (2022) combine Neural Fictitious Self-Play with KR-UCT to approximate Nash strategies in digital curling, focusing on large game-tree search in continuous action spaces and automatic generation of supervised data. Lee et al. (2018) present a self-play deep reinforcement learning framework for simulated curling that integrates continuous action search via kernel regression with high-fidelity simulation, achieving strong performance in an international digital curling competition without hand-crafted features. Our work shares the goal of high-quality continuous-action shot selection and, like Han et al. (2022) and Lee et al. (2018), uses KR-UCT, which was introduced for continuous-action MCTS with execution uncertainty by Yee et al. (2016). It differs in pairing search with an EfficientZero-style graph-transformer whose shared representation is trained by BYOL-style latent consistency, in planning directly against a calibrated physics simulator rather than a learned model, in optimizing robustness to a fitted Student-t noise model, and in recovering human priors by inverse dynamics from position-only match data.

Robotic and real-world curling. Beyond purely digital agents, several systems close the loop with the physical world. Won et al. (2020) introduce Curly, a curling robot powered by an adaptive deep reinforcement learning framework that incorporates temporal features to handle nonstationary ice conditions. Curly attains human-level performance in official matches against elite human teams, demonstrating that carefully adapted RL can narrow the sim-to-real gap in curling. Xiao et al. (2023) also emphasize deployment in real training environments, using perception modules and tailored MCTS to bridge from image observations to strategic recommendations. Compared to these systems, our work remains fully in simulation but focuses on the complementary problem of how to combine a learned graph-transformer representation with a calibrated simulator for tree search over full ends.

Local tactical optimization in curling. Several papers examine specific subproblems rather than full-game strategy. Ahmad et al. (2016) focus on the hammer shot, the final shot of an end, and investigate algorithms for selecting an optimal action in a continuous, low-dimensional, stochastic setting. They adapt a Delaunay triangulation method to this setting and show that it can outperform alternative algorithms and, under some conditions, even Olympic-level human performance in their simulator. Our approach instead targets multi-shot planning over an entire end, with continuous-action KR-UCT search against a physics simulator and a learned value, and, like Ahmad et al., optimizes directly in the continuous release-parameter space $[v, \theta, \omega, y_0]$ rather than over a discrete library of high-level shot types. Discretizing a continuous action space is the standard alternative, and the choice/granularity of discretization is itself consequential (Kroer & Sandholm, 2015). We avoid committing to a fixed grid, and instead let the mixture policy propose continuous candidates that search then evaluates under execution noise.

Physics-based strategy in other sports. The idea of using physics models or simulators to inform sports strategy is not unique to curling. The closest precedent is computational billiards, another continuous-control, execution-noise-dominated game: Archibald et al. (2009) analyse a winning computational billiards player and show that most of its strength comes from accounting for execution error when selecting among candidate shots, rather than from deeper search. Our noise-aware candidate ranking plays the same role here, over a longer horizon and against an adversary who also places stones. Fragkiadaki et al. (2015) learn visual predictive models of physics in a simulated billiards domain and use them to plan goal-directed actions via "visual imagination" over object-centric latent states. In ten-pin bowling, Ji et al. (2022) derive a rigid-body model for ball motion and use simulation to map initial conditions to strike probability, revealing that imperfect bowlers may benefit from particular targeting strategies that exploit the oil pattern to create additional "miss room". Conceptually we share these works' use of an explicit physics model for planning, but rather than *learning* the dynamics for use inside the planner, we treat a roughly calibrated curling simulator as the transition and learn only a representation that imitates it. We operate on object-centric stone positions rather than pixels and embed the simulator inside a continuous-action search loop for a multi-agent, zero-sum game.

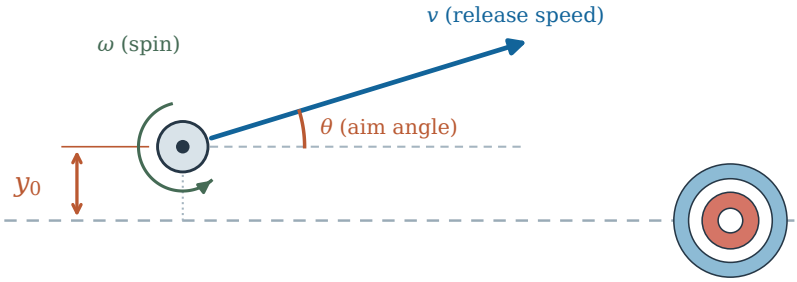


Figure 2: Release parameters. The lateral offset y_0 locates the stone relative to the sheet center line; v and θ set its initial velocity, and ω sets its spin. The button is shown at right.

World models and model-based reinforcement learning. Our method is also related to the broader literature on world models in reinforcement learning. DreamerV3 (Hafner et al., 2023) demonstrates that a single world-model-based algorithm can, with one configuration, achieve state-of-the-art performance across over 150 diverse tasks, including complex 3D environments, by learning a latent dynamics model and improving policies through imagination. Alonso et al. (2024) introduce DIAMOND, a diffusion-based world model for Atari that shows how modeling high-fidelity visual details via diffusion can translate into improved downstream policy performance. In our setting we draw on these ideas for representation learning, namely an action-conditioned latent-dynamics head and a BYOL-style latent-consistency objective (Grill et al., 2020) in the EfficientZero spirit (Ye et al., 2021), but, unlike Dreamer and DIAMOND, we do *not* roll out the learned model for planning. We operate in a low-dimensional, object-centric state space and use a roughly calibrated physics simulator as the transition, reserving the learned dynamics for shaping the shared graph-transformer representation.

Planning with learned models. Finally, our planning setup is inspired by MuZero (Schrittwieser et al., 2020), which combines tree search with a learned model to achieve superhuman performance in Atari, Go, chess, and shogi without explicit knowledge of the game rules. MuZero learns a latent dynamics model that produces policy, value, and reward predictions suitable for search. In contrast, we deliberately do *not* plan in the learned latent model. MuZero learns dynamics, reward, value, and policy expressly for in-model search, whereas we borrow only its shared-representation and latent-consistency machinery and run the actual search against a calibrated simulator with a continuous-action KR-UCT tree (Yee et al., 2016) and an inverse-recovered human prior. Furthermore, to find meta-Nash strategies, we design a population-based framework roughly following Lanctot et al. (2017), populated with our search-distilled policies. Relative to prior curling work, which either relies on a hand-coded simulator with classical search or on model-free RL in continuous action spaces, our contribution is to combine an EfficientZero-style graph-transformer representation with simulator-grounded planning and execution-noise-robust, search-distilled policies for mixed doubles curling.

3 Problem Setting

The board state $s = (P, m)$ contains twelve fixed stone slots, six per team. Their normalized two-dimensional coordinates are $P \in [0,1]^{12\times 2}$, and $m \in \{0,1\}^{12}$ is a live mask indicating whether each slot currently contains an in-play stone. Coordinates are used only for live slots. The graph encoder augments these state variables with team identity and deterministic geometric features described in Section 4 and Appendix D. Context $c \in \mathbb{R}^3$ records progress through the end, throwing order, and the current team's stone block. The continuous intention $a = [v, \theta, \omega, y_0]$ specifies release speed, aim angle, spin, and lateral release offset. The simulator perturbs this intention with execution noise before applying the throw (Figure 2). Full definitions and bounds are in Appendix B.

The horizon $h \in \{1, \ldots, 10\}$ is the number of throws remaining, including the current throw. Throws alternate, so odd h gives the team-to-move the hammer and even h gives it to the opponent. An episode is one complete end. Its terminal reward is the signed score margin under mixed-doubles rules. All simulated play enforces the free-guard-zone rule (Appendix N).

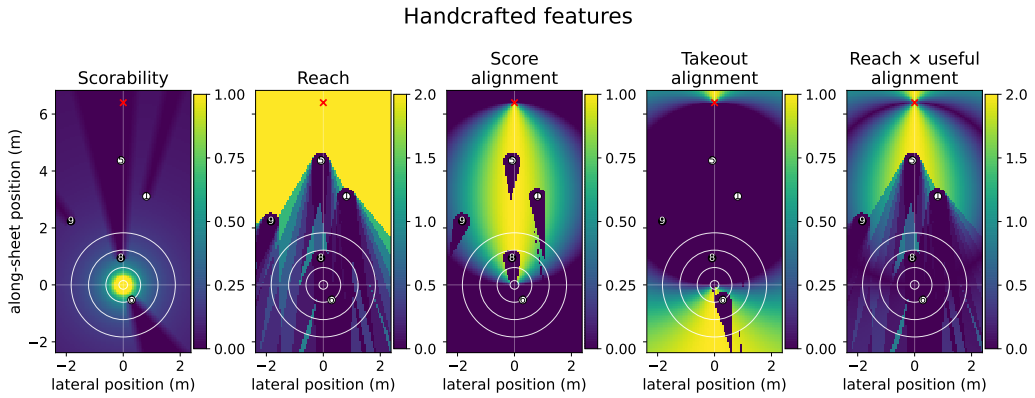


Figure 4: Handcrafted edge features used by the final model. The numbered stones remain fixed while the opponent target varies across the sheet. SCORABILITY measures visibility from the button, REACH measures curl-path accessibility, and SCORE ALIGNMENT/TAKEOUT ALIGNMENT measure directional alignment toward scoring or removal. Yellow denotes larger values.

For a fixed simulator and execution-noise model, we seek a population whose empirical equilibrium is increasingly difficult to exploit. The learned policy should also support human analysis: it exposes distinct shot modes, search trajectories, and a value distribution for any board. We therefore measure both strength of play in the simulator and whether expert curlers judge its decisions preferable to recorded human ones. These records come from elite international mixed-doubles competition, including the Olympic Games and World Mixed Doubles Curling Championships, so the recorded actions are drawn from national-team-level play rather than recreational or lower-tier competition.

4 A GRAPH-TRANSFORMER MODEL WITH SIMULATOR-GROUNDED SEARCH AND ITERATIVE BEST-RESPONSE TRAINING

The encoder $E(s, c)$ maps the state graph to $h \in \mathbb{R}^{256}$. Three heads share this representation (Figure 3):

$$\underbrace{\pi(a \mid h)}_{\text{shot modes}}, \qquad \underbrace{V(h) = (\mu_V, \sigma_V)}_{\text{score distribution}}, \qquad \underbrace{G(h, a)}_{\text{next latent}}.$$

The policy proposes several types of shot, search chooses among them, and the value head scores simulated continuations. The latent head receives any action and predicts the encoding of the simulator's next board. It provides auxiliary representation-learning supervision. It is never used for imagined planning: all candidate trajectories come from the explicit simulator, so their physical causes and parameter settings can be inspected.

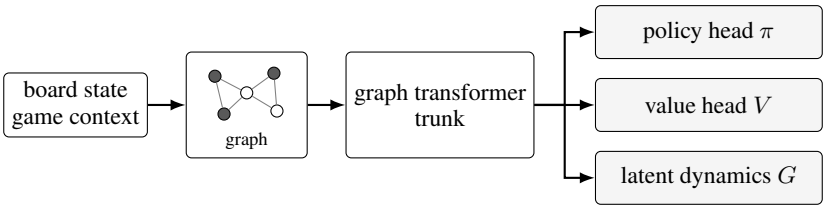


Figure 3: State and game context form a graph. A shared graph transformer produces a multimodal shot policy, a probabilistic position value, and an action-conditioned next-state representation.

Shared graph-transformer trunk. The graph has 12 stone nodes, nine fixed landmark nodes, and one global context token. Dense edges encode relative position, team identity, visibility from the button, curl-arc reachability, and whether a contact can send a stone toward the button or out of the house. These geometric features are deterministic and recomputed for every state. Figure 4 visualizes several of the feature fields used by the transformer; Appendix A gives an additional example.

The trunk has eight edge-biased transformer layers, width 256, four attention heads, and dropout 0.1. Edge features affect both attention logits and values. The final global token is the shared

latent h. The deployed model has 7,180,550 trainable parameters, excluding the non-gradient target encoder used for latent consistency. Horizontal reflection and team-slot exchange provide symmetry augmentation. Appendix D gives the full graph definition.

Our architecture has three heads. The policy head is a 16-component full-covariance Gaussian mixture over standardised release actions. The value head predicts a Gaussian distribution over realized signed end margins. The latent head predicts the next encoded state from the current latent and action; it trains the shared representation but is never rolled forward during search. Appendix D gives the head parameterisations.

Simulator and Recovering Human Action. A batched JAX simulator models friction, curl, spin decay, and stone contact. Human throws are recovered by inverse-solving with a separately tuned contact and ice preset (Appendix C); play, search, and evaluation use one fixed parameter set for all comparisons. Search uses its physical transitions directly; bounds and numerical details are in Appendix B.

Execution noise is explicit in collection and evaluation. We add diagonal Student-t noise ($\nu = 5$) to $[v, \theta, \omega, y_0]$. Speed and angle scales follow the Draw Shot Challenge fit reported by Yee et al. (Yee et al., 2016); spin and offset scales are empirical because no comparable public measurements were available. Noise affects both candidate ranking and the realized throw. Appendix G reports every scale and the seed protocol.

The match records contain boards before and after each shot but not release parameters. A Cross-Entropy Method inverse solver (De Boer et al., 2005) searches for an action whose simulated next board matches the observation. Low-residual solutions initialize the policy and encoder; observed end scores initialize the value head. Later response rounds train on simulated play, not continued human imitation. Appendix C defines the matching loss and quality filter.

Continuous-action search and target generation. At each learner turn, four-ply KR-UCT proposes an action using 16,000 simulator calls (Yee et al., 2016). Opponent nodes sample the fixed opponent policy; after the tree depth, rollouts continue to the end. An independent gate compares the proposal with the current policy action using eight matched-seed continuations. We accept the proposal only when its mean gain is positive and its paired statistic exceeds 0.5; otherwise the current action becomes the policy target. At deployment, the policy samples 48 shots, simulates eight noisy executions of each, and selects the highest mean-value intention for a fresh execution draw. Appendix E gives the tree selection rule.

Targets and losses. Each simulated trajectory supplies three forms of supervision. On learner turns, the policy target a^* is the accepted tree proposal or the current policy fallback. Every state receives the realized terminal score R from the appropriate player's perspective. Each state–action transition also provides the simulator's true next state s'. The realized score is a noisy Monte Carlo target for the value head; it does not directly choose the policy label a^*. The core objective is

$$\mathcal{L}_{\text{core}} = \lambda_\pi \big[-\log \pi(a^* \mid E(s,c)) \big] + \lambda_V \mathcal{L}_V(V(E(s,c)), R) + \lambda_C \mathcal{L}_{\text{cons}}. \tag{1}$$

The value loss combines squared error on μ_V with Gaussian NLL, so the model learns both expected score and uncertainty. The consistency loss compares the predicted next latent with a stop-gradient encoding of the simulator's next board:

$$\mathcal{L}_{\text{cons}} = d(G(E_\theta(s,c), a), \operatorname{sg} E_{\bar{\theta}}(\text{Sim}(s,c,a), c')), \quad \bar{\theta} \leftarrow 0.99\bar{\theta} + 0.01\theta, \tag{2}$$

where d is the SimSiam-style prediction loss and $E_{\bar{\theta}}$ is an exponential-moving-average target encoder (Grill et al., 2020). Figure 6 summarises the two branches. Thus learned dynamics shapes the representation, while the interpretable simulator alone supplies planning transitions.

4.1 Human initialization and iterative response training

Human matches initialize π_{H} and the value representation. At round r, we solve the current population's empirical zero-sum game, sample opponents from its meta-Nash mixture μ_{r-1}, and collect full ends in both throwing roles. Learner turns use the paired-gated search target of Section 4; realized scores and simulator transitions train the value and latent heads. A new policy enters the population only after an independent head-to-head gate. We stop after three rounds. The final response used 480 simulated ends, producing 2,400 learner policy targets; training and checkpoint details are in Appendix F.

Table 1: Final response policy π_3 against the preceding iterations of response policies under stochastic execution. Uncertainty is one standard error over 40 evaluation cells.

Opponent	win	dScore/end	t
π_1	0.523	$+0.169 \pm 0.034$	4.99
π_2	0.543	$+0.120 \pm 0.027$	4.40

Table 2: Empirical-game progression across the three response policies. The first two response scores pool two independent evaluations. We run three rounds: each is a full search-based collection and training cycle, so the count is set by compute rather than by a convergence criterion.

Round	policy	$\max \mu_{r-1}$	vs. incumbent	vs. μ_{r-1}
1	π_1	1.000	$+0.079 \pm 0.024$	$+0.079 \pm 0.024$
2	π_2	1.000	$+0.025 \pm 0.023$	$+0.025 \pm 0.023$
3	π_3	0.652	$+0.169 \pm 0.034$	$+0.143 \pm 0.020$

5 EXPERIMENTS

We describe the data, metrics, and baselines, then report empirical-game results, the language-model arena, value calibration, and the expert study.

Data. We use shot-by-shot records from international mixed-doubles competitions. Competition-level splits prevent boards from the same event appearing in both training and evaluation. Human action labels are inverse-solved as described in Section 4. Response-round data comes from complete simulated ends and retains the generating policy, opponent, opening, horizon, random seed, and search decision in each shard manifest.

Metrics. Our primary metric is score differential per end (dScore) under stochastic execution. We also report win score (win $= 1$, draw $= 0.5$, loss $= 0$) and the separate win/draw/loss frequencies. Every matchup uses both throwing orders and all horizons. The statistical unit is an independently seeded horizon–shard cell after averaging its two throwing orders, not an individual simulated throw. We report the mean, standard error, and $t = \text{mean}/\text{SE}$. Human-data policy NLL, value MSE, Gaussian NLL, and interval coverage are diagnostics, not measures of playing strength. Promotion requires positive dScore with $t \geq 2.1$.

Baselines. We compare each response with its incumbent, training mixture, earlier responses, and human initialization. Search ablations compare the raw policy, deployed selector, tree proposal, and paired-gated teacher; the protocol is in Appendix K. Five language-model agents use the same text-state interface, both throwing orders, execution noise, and fixed tool budgets (Appendix H). Claude Opus 5 and GPT-6-Astra are the frontier baselines; the other three are hosted open-weight models. Published curling agents use incompatible simulators, action spaces, or rules, so their reported scores are not direct head-to-head baselines.

5.1 EMPIRICAL-GAME RESULTS

Table 1 reports the final response policy, π_3, against the two earlier response policies retained for presentation. Each matchup contains 2,516 simulated ends in 40 evaluation cells. Table 2 summarises the three response rounds.

Against the previous meta-Nash, π_3 scores $+0.1433 \pm 0.0195$ points per end ($t = 7.33$); its matchups against both earlier responses pass the pre-specified $t \geq 2.1$ gate (Tables 1–2). The first two restricted games were nearly pure; before round 3, π_2 held 65.2% of mixture weight. After round 3, the empirical solution placed 99.996% on π_3. This is strength within the measured population, not a continuous-game equilibrium guarantee.

5.2 LLM, PREDICTION, AND EXPERT EVALUATION

Table 3 reports standard ends against five language-model agents: Claude Opus 5 and GPT-6-Astra as frontier baselines and three hosted open-weight models. All use a shared text-state protocol, both throwing orders, and the same execution noise. Power-play ends are reported separately because the agent may call one whenever it holds hammer, while π_3 follows a fixed rule; pooling the two strata would confound that access difference (Appendix H). Win rates are end-wise. A *preview* lets an agent submit a candidate shot to the arena before committing to it and see what the simulator does with it: the solver's achieved error and one noisy realization of the resulting board. We offer it because the comparison is otherwise lopsided. Our policy was trained inside this simulator and

Table 3: Standard-end head-to-head results against language-model agents. dScore is from π_3's perspective; $\pm$ is one standard error. Win counts blanks as half. Each row pools the zero-preview and two-preview conditions.

Agent	standard ends	win	dScore/end
Claude Opus 5	80	0.500	$+0.050 \pm 0.269$
GPT-6-Astra	61	0.541	$+0.443 \pm 0.298$
`gpt-oss-120b`	15	0.667	$+1.000 \pm 0.632$
`gpt-oss-20b`	15	0.600	$+0.933 \pm 0.636$
`qwen3.8-27b`	20	0.650	$+0.950 \pm 0.489$
open-weight (pooled)	50	0.640	$+0.960 \pm 0.326$

Table 4: Balanced power-play ends against Claude Opus 5, the frontier agent that is level with π_3 on standard ends: 25 held by each side. The final row is π_3's overall head-to-head result.

Power play held by	ends	win	dScore/end
π_3	25	0.920	$+2.320 \pm 0.386$
Claude Opus 5	25	0.280	-1.120 ± 0.371
π_3 dScore, all 50 ends	50	0.600	$+0.600 \pm 0.268$

ranks 48 candidates by simulating each of them, whereas a language model meets the physics for the first time at test time and must predict the outcome of a shot it has never seen executed. A small preview budget is the closest we can come to letting it check a shot the way our own policy does, without granting it the full candidate sweep. Each row pools the zero-preview and two-preview conditions, which we combine because the budget made no reliable difference: against Claude Opus 5 the policy scored $+0.056 \pm 0.320$ over 54 standard ends with no previews and $+0.038 \pm 0.501$ over 26 with two previews. GPT-6-Astra behaves the same way across conditions: $+0.500 \pm 0.345$ over 48 standard ends with no previews and $+0.231 \pm 0.601$ over 13 with two. Its pooled standard-end margin, $+0.443 \pm 0.298$, is nominally larger than Claude Opus 5's but not distinguishable from it ($+0.444 \pm 0.470$, $t = 0.95$). The two frontier agents do differ in how they play: GPT-6-Astra drew on 79% of its throws and issued no raw-parameter shots, whereas Claude Opus 5 drew on 55%, removed stones four times as often, and issued raw parameters on a quarter of its throws.

Across 50 standard ends against hosted open-weight agents, π_3 scores $+0.960 \pm 0.326$ points per end and wins 64%. The two frontier agents differ. π_3 is ahead of GPT-6-Astra, $+0.443 \pm 0.298$ over 61 standard ends, though not significantly, and level with Claude Opus 5: $+0.056 \pm 0.320$ without previews (54 ends) and $+0.038 \pm 0.501$ with two previews (26 ends). Because Claude Opus 5 is the stronger of the two and cannot be separated from π_3 on standard ends, we compare them on power-play ends.

In the balanced 50-end power-play test (Table 4), π_3 scores $+0.600 \pm 0.268$ per end. It gains $+2.320 \pm 0.386$ when holding the power play and concedes 1.120 ± 0.371 when Claude Opus 5 holds it; the conversion difference is $+1.200 \pm 0.536$ (permutation $p = 0.035$). A one-power-play-per-side eight-end weighting gives $+0.19 \pm 0.21$ per end. Protocol and additional breakdowns are in Appendix I.

On held-out human ends ($n = 3{,}851$ states from the competition test partition, which is never used for checkpoint selection), value MSE was 2.69 (RMSE 1.64 against an outcome standard deviation of 1.85), mean Gaussian NLL was 1.91, and the nominal 90% interval covered 87.4% of outcomes (95% bootstrap CI $[86.4, 88.4]$). The 50% and 80% intervals covered 51.0% and 78.3%, so the predictive distribution is well calibrated near the median and mildly overconfident in the tails. An ablation removing *policy search* at deployment, drawing a single shot from the same policy head instead of ranking 48 noise-averaged candidates with the value head, costs 1.24 ± 0.11 dScore per end ($t = 11.7$; 400 ends over 200 held-out roots, all horizons, both throwing orders, Appendix K). The gap widens with stones remaining, from $+0.13$ with one stone left to $+1.95$ at the start of an end, as one would expect of an advantage that compounds. We also ablate the two auxiliary training objectives by retraining the final response with the corresponding loss weight set to zero, holding initialization, replay data, architecture, optimization budget, and checkpoint selection fixed. Removing latent consistency changes held-out policy-target NLL from 5.2546 to 5.2650 and value MSE from 3.683 to 3.688; removing the probabilistic value objective leaves policy-target NLL at 5.2546 and moves value MSE to 3.684 (Appendix K). This isolates the contribution of each objective in the final response round rather than retraining the entire three-round response ladder.

Two U.S. Olympic Curling Team staff members completed a blind paired test on 30 held-out recorded positions, yielding 60 judgments. Each pair contained the recorded human action and π_3's action for the same state, rendered identically and in randomized order. Asked which shot looked more optimal, the reviewers selected π_3's action in 26 of 60 judgments (43.3%; Wilson 95% CI $[31.6, 55.9]$). Because both reviewers judged the same 30 positions, we cluster by position, which gives $43.3 \pm 7.5\%$ ($t = -0.89$ against chance). The test therefore detects no preference in either direction. The two reviewers agreed on 20 of 30 positions ($\kappa = 0.32$) and individually selected π_3 in 40.0% and 46.7% of trials. Free-text feedback from the same staff was more positive than the forced choice: the policy was described as "making a lot of good shot decisions", with "only a couple" judged dubious, specifically risky tap decisions on opening throws, which would not be seen in competition. The reviewers attributed this partly to the simulator's limited curl, which matters most for opening shots, so it may reflect the environment rather than the policy.

6 LIMITATIONS

All strategic claims are conditional on one simulator and one action space. The simulator omits changing ice, sweeping, and player-specific execution, and only its speed and angle noise scales have an external empirical source. The inverse solver estimates one plausible human intention rather than a posterior over intentions. Our empirical meta-Nash covers a finite learned population and cannot rule out an unseen response. The LLM comparison covers five model snapshots on one evaluation date and the expert study, in the round reported here, contains two reviewers, 30 paired positions, and 60 individual judgements so neither establishes performance against all general-purpose agents or curlers. The LLM protocol could benefit from further work in balancing the playing ground for fair evaluation. In particular, the power-play advantage over Claude Opus 5 rests on 50 isolated single-end matches, and we did not run that experiment against GPT-6-Astra, and a full-match advantage is not established. The expert evaluation protocol also has a confound in displaying each option's post-shot board, such that realization luck is visible and mixes into a judgement meant to be about shot selection. Blind preferences in any case measure perceived shot quality, not real-world playing strength.

7 CONCLUSION

We combine a multimodal shot policy, probabilistic position value, and action-conditioned latent model in one graph transformer. Human data provides the initial representation; simulator search and realized self-play returns then improve it through numbered best-response rounds against an empirical meta-Nash. The final response is positive against both preceding response policies and the previous meta-Nash under stochastic execution. On standard ends it scored $+0.96 \pm 0.33$ dScore/end against hosted open-weight language-model agents, was ahead of GPT-6-Astra, and played even with the stronger frontier agent, Claude Opus 5, which it outscored on balanced power-play ends ($+0.60 \pm 0.27$ per end), and U.S. Olympic Curling Team staff, asked which shot looked more optimal, showed no significant preference between its actions and recorded human actions across 60 blind judgments on 30 paired positions. Together, these results show that search over a learned probabilistic value, trained by best response against an empirical meta-Nash, produces a curling policy that general-purpose reasoning does not reach by reasoning about the state alone.

AI USE STATEMENT

In this work, we used generative AI tools for code generation and debugging, research ideation, literature discovery, and manuscript drafting and editing. We did not use generative AI to fabricate experimental observations: reported quantitative results were produced by the documented simulator and evaluation pipeline. LLM play is itself an explicitly labelled experimental baseline. We manually verified references, tested AI-assisted code, and reviewed all AI-assisted text and analysis. We take responsibility for the final content of this work, including material produced with the aid of generative AI.

ETHICS STATEMENT

The quantitative study uses existing sports records and simulation. We do not release personally sensitive information and use match data according to its available terms. The expert study used informed participation. We separated design feedback from formally analysed responses and attributed no quotation without explicit permission.

REPRODUCIBILITY STATEMENT

The main text specifies the state, action, objective, search operator, target construction, and evaluation protocol. The appendix gives the simulator and free-guard-zone rules, action bounds, inverse-action solver, architecture, optimization settings, execution-noise distribution, and compute setup. Each reported head-to-head comparison uses both throwing orders; paired random-number streams are used where the compared alternatives share stochastic events. We report the number of independent evaluation cells together with score-differential uncertainty. Public match records are identified by competition and split. Derived inverse actions and simulator-generated search targets include scripts and provenance manifests for regeneration where redistribution terms permit. LLM baseline artifacts include dated model identifiers, prompts, tool schemas, sampling settings, and raw-output audit records.

REFERENCES

Zaheen Farraz Ahmad, Robert C Holte, and Michael Bowling. Action selection for hammer shots in curling. In *IJCAI*, pp. 561–567, 2016.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

Christopher Archibald, Alon Altman, and Yoav Shoham. Analysis of a winning computational billiards player. In *IJCAI*, volume 9, pp. 1377–1382, 2009.

Michael Brunner and Kaspar Riesen. Comparing learning methods to enhance decision-making in simulated curling. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 156–165. Springer, 2024.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Yuntao Han, Qibin Zhou, and Fuqing Duan. A game strategy model in the digital curling system based on nfsp. *Complex & Intelligent Systems*, 8(3):1857–1863, 2022.

Simon Ji, Shouzhuo Yang, Wilber Dominguez, and Cacey Bester. Using physics simulations to find targeting strategies in competitive bowling. *arXiv preprint arXiv:2210.06753*, 2022.

Christian Kroer and Tuomas Sandholm. Discretization of continuous action spaces in extensive-form games. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pp. 47–56, 2015.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Kyowoon Lee, Sol-A Kim, Jaesik Choi, and Seong-Whan Lee. Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling. In *International conference on machine learning*, pp. 2937–2946. PMLR, 2018.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Dong-Ok Won, Klaus-Robert Müller, and Seong-Whan Lee. An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions. *Science Robotics*, 5(46):eabb9764, 2020. doi: 10.1126/scirobotics.abb9764. URL https://www.science.org/doi/abs/10.1126/scirobotics.abb9764.

Qian Xiao, Zongmin Li, Xiangdong Wang, Yujie Liu, Yachuan Li, Chaozhi Yang, and Feimo Li. Policy decision of curling in real competition scenes. *Complex & Intelligent Systems*, 9(3):3301–3312, 2023.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in neural information processing systems*, 34:25476–25488, 2021.

Timothy Yee, Viliam Lisỳ, Michael H Bowling, and S Kambhampati. Monte carlo tree search in continuous action spaces with execution uncertainty. In *IJCAI*, pp. 690–697, 2016.

A GEOMETRIC EDGE FEATURES

The model uses five deterministic geometric edge features. SCORABILITY is the target's unoccluded angular width from the button (radians), measuring scoring visibility. REACH is the feasible curl-path fraction multiplied by the normalized range of feasible sampled curvatures (dimensionless, $[0, 2]$), measuring route diversity. SCORE ALIGNMENT and TAKEOUT ALIGNMENT are feasibility-gated positive directional cosines (dimensionless, $[0, 1]$), aligning the incoming direction with, respectively, the target-to-button and button-to-target directions. They encode draw alignment and outward-hit alignment. The final feature, REACH $\max(\text{SCORE ALIGNMENT}, \text{TAKEOUT ALIGNMENT})$, combines accessibility with the more useful outgoing direction and lies in $[0, 2]$.

Figure 5 shows these features on a more crowded board configuration.

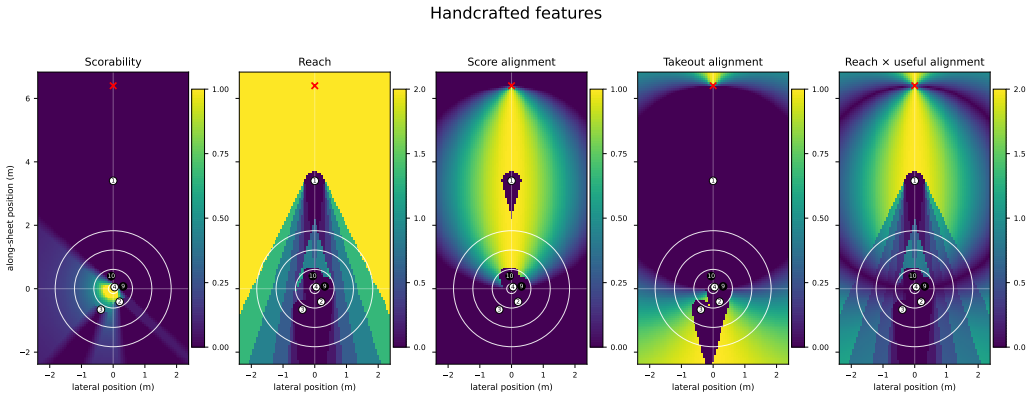


Figure 5: The five handcrafted edge features on a more crowded board. The numbered stones remain fixed while the opponent target varies across the sheet; yellow denotes larger values.

B CURLING PHYSICS SIMULATOR

The transition $s' = \text{Sim}(s, c, a)$ is a JAX physics engine integrated by semi-implicit Euler (macro step $dt = 0.02\,\text{s}$, two substeps, up to 1,500 steps, stopping when all stone speeds fall below $v_{\text{stop}} = 0.03\,\text{m/s}$). The thrown stone is released at $[-d_{\text{hog-tee}}, y_0]$ with $d_{\text{hog-tee}} = 28.35\,\text{m}$ (full sheet) and initial velocity $v\,[\cos\theta, \sin\theta]$ and spin ω. Dynamics combine: (i) linear ice friction with deceleration $a_{\text{linear}} = 0.11\,\text{m/s}^2$ opposing motion; (ii) a spin-induced curl lateral acceleration $a_{\text{curl}} = k_{\text{curl}}\,\omega\,\hat{v}^{\perp}\,s_{\text{eff}}$ with $k_{\text{curl}} = 0.10$ and a speed-saturated factor $s_{\text{eff}} = c_{\text{cap}}\tanh(v/c_{\text{cap}})$, $c_{\text{cap}} = 2.5\,\text{m/s}$; (iii) exponential spin decay at rate $\gamma_{\text{spin}} = 0.15\,\text{s}^{-1}$, with additional damping $c_{\text{spin-contact}} = 80$ during contact. Stone-stone interaction uses smooth penalty contact forces with stiffness $k_{\text{penalty}} = 2.5\times10^4\,\text{N/m}$, damping $c_{\text{damp}} = 220\,\text{kg/s}$, and a force clamp of $f_{\max} = 2\times10^4\,\text{N}$ to prevent stiff-spring blow-ups; stone mass is $m = 19.96\,\text{kg}$ and stone radius is $r = 0.145\,\text{m}$. Initial overlaps from data quantisation are removed by an iterative projection. Sheet walls are disabled; the in-play check is purely geometric. The action ranges (full sheet, realism-fixed) are

$$v \in [2.20,\ 3.01]\,\text{m/s}, \qquad \theta \in [-0.1038,\ 0.1038]\,\text{rad},$$
$$\omega \in [-7.0,\ 7.0]\,\text{rad/s}, \quad y_0 \in [-0.23,\ 0.23]\,\text{m},$$

and serve as a hard *search-space limit* during target generation: KR-UCT widening and one-ply sampling never propose outside these bounds, and the inverse solver also samples within them. The board has 12 stones (six per team); two are pre-placed and ten are thrown per end; the new stone fills the thrower's next free slot. The simulator is exposed as a batched JAX function and is the only transition called during search, target generation, and evaluation.

C INVERSE-ACTION SOLVER

Given a pre-state and an observed post-state, we recover the release parameters by minimising a position-matching loss with the cross-entropy method (CEM) (De Boer et al., 2005). Candidate

actions in normalized $[0, 1]^4$ are sampled from a Gaussian, simulated forward, and scored by a team-ownership-aware, permutation-invariant assignment loss: matched live stones contribute squared position error, while unmatched targets and spurious extra in-play stones incur capped penalties ($\leq 4\,\mathrm{m}^2$ each), so the loss correctly handles stones knocked out of play. CEM runs in two stages, each with EMA-smoothed mean and variance and a variance floor: a coarse stage on a lower-resolution integration (population 800, 25 generations, elite fraction 0.2, early stop at residual $\leq 0.5\,\mathrm{m}^2$), then a refinement stage initialised from its result (population 800, 80 generations, elite fraction 0.3, early stop at residual $\leq 0.1\,\mathrm{m}^2$). Both stages simulate with a contact and ice preset tuned for inverse solving (curl coefficient 0.12, linear friction 0.10, spin decay 0.12, with tangential contact friction), which differs from the default parameters used for play, search, and evaluation (0.10, 0.11, 0.15, no tangential friction). We retain solutions that converge (`solver_ok`) with refined residual at most $0.08\,\mathrm{m}^2$ to form the "realistic" inverse dataset. Release parameters are z-standardised by the dataset mean/std (μ_a, σ_a), stored with the checkpoint and used to map policy outputs back to physical actions.

D Architecture and Feature Details

We use the curling-aware graph-feature configuration that the warm-start encoder was trained with; the same configuration is used by our standalone value baseline, so the two are directly comparable. The exact feature set is pinned in the implementation; an additional visualization is given in Appendix A.

Nodes (22). 12 stone-slot nodes (alive iff the slot holds an in-play stone), plus 9 *landmark* nodes: the button centre at raw $(750, 800)$, three release-area reference points across the nearer hog line (left, centre, right) at $\approx 6.4\,\mathrm{m}$ forward of the button along the centre line and at lateral offsets $\{-1.22, 0, +1.22\}\,\mathrm{m}$, and five behind-house takeout-boundary points placed $\approx 2.4\,\mathrm{m}$ behind the button (just past the back of the house) at lateral offsets $\{-1.83, -0.91, 0, +0.91, +1.83\}\,\mathrm{m}$. The 22nd node is a *global token* that is prepended to the node tensor at encode time and whose features are the condition c projected to the hidden dimension; the latent h is read out as the output embedding of this token. Node features are 6-dimensional: $[x, y, \text{team_indicator}, \text{is_live}, \text{is_landmark}, \text{spare}]$, where `team_indicator` is $0/1$ for stone slots and 0 for landmarks; the trailing scalar is unused.

Edges (9-d features). 4 base pairwise scalars $[\Delta x, \Delta y, \text{distance}, \text{same_team_indicator}]$ plus 5 curling-specific scalars: SCORABILITY (unoccluded angular span of the target seen from the button, computed from board geometry at the current step and broadcast as a node feature into each incident edge); REACH (curl-arc reachability on release-source edges, pairwise angular reachability otherwise); SCORE ALIGNMENT (outgoing-direction compatibility with drawing to the button); TAKEOUT ALIGNMENT (outgoing-direction compatibility with ejecting an opponent through the behind-house corridors); and the composite REACH $\cdot \max($SCORE ALIGNMENT, TAKEOUT ALIGNMENT$)$. Edges are not pruned in the current configuration. The composite scalars are deterministic functions of the live board state rather than learned, and they recompute every step. They are projected into a per-head bias and a value modulation at each transformer layer, so the attention is *edge-biased* rather than purely token-based.

Transformer. Eight layers, hidden width 256, four attention heads, and dropout 0.1; the readout is the global-token embedding. The deployed network has 7,180,550 trainable parameters, excluding the moving-average target encoder.

Heads. The policy head emits a $K = 16$ full-covariance Gaussian mixture over standardised actions, the value head emits a Gaussian $(\mu_V, \log \sigma_V^2)$, the dynamics head is a residual MLP $G(h, a_{\text{box}})$. A BYOL-style target encoder $E_{\bar{\theta}}$ ($\tau = 0.99$, no gradient) supplies the consistency target.

Warm start. The trunk loads the matched modules (`global_token`, `node_proj`, `cond_proj`, and the transformer layers) from the human-prior policy checkpoint. The policy head is loaded by default (and then refined by search distillation). The value head loads the mean and log-variance projection submodules from a separately trained standalone value model that uses the same graph configuration. The BYOL target encoder is synced to the trunk before training begins.

Augmentation. At training time we apply, independently with probability 0.5 each, a horizontal flip (mirror stone x across the centerline; negate the angle, spin, and lateral-offset components of

all action targets) and a team-slot swap (exchange slots 1–6 with 7–12 and flip stone_block). These keep value and outcome targets invariant while doubling the effective data.

E Search Implementation Details

Inside the tree, evidence is shared between nearby actions by kernel regression whose bandwidth is measured in units of the execution-noise standard deviation, so each open action a has a kernel-weighted count $W(a) = \sum_b K(a,b)\, n(b)$ and a kernel-regressed value $V(a)$. Selection maximises $V(a) + c_{\mathrm{puct}}\, p(a) \sqrt{\sum_b W(b)} \,/\, (1 + W(a))$, so exploration is driven by *visitation*, as in PUCT, with the kernel preventing two nearly identical shots from counting as independent evidence; the value head's predicted σ_V is not used as an exploration bonus. Progressive widening opens one further candidate only when the leading half of the open actions each carry kernel-weighted evidence above a fixed threshold, which keeps the tree from widening on noise.

F Training and Compute

The current final response collected 480 complete ends: 160 standard openings, 159 right power plays, and 161 left power plays. This produced 4,800 transition records and 2,400 learner-turn policy targets. The paired test accepted 885 tree proposals and kept 1,515 reference-policy actions. A fixed shard split yielded 3,840 training and 960 validation transitions.

Each learner decision used a four-ply, 16,000-call opponent-model tree with eight children per non-root node and a root output cap of 64. Eight matched continuations screened its top action. The first explicit opponent reply approximated deployed selection with 16 candidates and two noise samples; actual opponent turns used the deployed selector. Collection took about 54.1 hours with eight workers distributed across four NVIDIA A10Gs.

Training mixed the 3,840 response records with 85,676 value records. The effective draw proportions were 88.9% response data and 11.1% value data. We used PyTorch, AdamW, batch size 256, 90,000 sampled examples per epoch, gradient-norm clipping at 1.0, symmetry augmentation, and learning rate 1.5×10^{-5} for inherited parameters. Training ran on one A10G, took about 45 minutes, and stopped after five new epochs; the first new epoch minimised held-out policy-target NLL while satisfying the pre-specified value MSE guard of 5.50. Collection, training, and evaluation use separate seed namespaces and write machine-readable manifests.

G Execution-Noise Model

Execution noise is diagonal Student-t with $\nu = 5$. Its *scale* parameters in action order $[v, \theta, \omega, y_0]$ are 0.0095 m/s for speed, 0.00116–0.00365 rad for the speed-dependent angle term, 2.0 rad/s for spin, and 0.015 m for lateral offset. Because a Student-t with $\nu = 5$ has variance $\frac{\nu}{\nu-2}$ times its squared scale, the realized standard deviations are $1.29\times$ these values: measured over 10^5 draws from the sampler, 0.0122 m/s, 0.0023 rad, 1.98 rad/s, and 0.0148 m. The tails matter in practice as well as the scale: the 99th percentile of the speed perturbation is 0.038 m/s, roughly three times its standard deviation. Speed and angle come from the Draw Shot Challenge fit reported by Yee et al. (Yee et al., 2016); spin and offset are empirical settings. Standard draws are capped at magnitude 8 for numerical safety, but realized actions are not clipped back to the intention bounds. Selection noise and realized execution use independent, recorded random streams.

H LLM Arena Baseline Protocol

We evaluate general-purpose language models through the same arena under a *text-state* protocol. At each turn the agent receives a plain-text record containing every stone's coordinates, team to move, horizon, hammer, opening mode, power-play availability, action bounds, and the scoring and free-guard-zone rules; the same record a human sees rendered in the interface. The agent returns a shot as a constrained JSON object, either as raw (v, θ, ω, y_0) or as one of the solver-backed intents (draw to a target, contact at a point with a weight class, or move/remove an identified stone), which

the arena's inverse solver converts to physical parameters and reports the achieved error for. We use a text rather than image observation because it is the modality all evaluated agents share, and because rendering introduces a perception confound that is not the object of study: the text record is information-complete with respect to the simulator state.

The action interface is shared across language-model agents but is not identical to π_3's trained support. Solver-backed intents remain within the trained limits, whereas explicit raw actions may use the arena's expanded spin and lateral-offset limits. We retain this asymmetry across the agentic replications rather than describe the physical action bounds as matched.

The GPT-6-Astra replication uses the `gpt-6-astra` identifier at medium reasoning effort in an agent harness. A fresh player context handles each pair of eight-end matches, one per seat. Its access is restricted by construction: the transport helper is the only channel to the arena, and the player context is given no filesystem access, so it can read neither the simulator source and configuration nor the match records of other runs. Its knowledge of the physics comes only from the responses to its own throws. The replication played 58 regulation ends with no previews and 16 with at most two previews per turn. Initial match seeds and hammer assignments match the corresponding Claude runs, but different actions and shared server-side sampling mean that entire trajectories are not paired. The transport helper logs requests, enforces preview budgets, and withholds champion value estimates except in permitted previews. After three rejected commits on a turn, the fallback is a raw centre draw. Model temperature and token counts are not reported when unavailable from the agent harness.

A tool-enabled condition additionally permits a fixed budget of simulator previews per turn, in which the agent submits a candidate shot to the arena's `solve` endpoint and receives the solver's achieved error and the predicted resulting position before committing. A preview returns one execution realization, not an expectation. The zero-preview and tool-enabled conditions are reported separately.

The Claude Opus 5 players at max reasoning effort are unfairly over-credited in our paper, because they were not restricted the same way. They run in an agent harness with shell access to the machine that hosts the arena, so nothing mechanically prevents them from reading the simulator source or the match records of other runs, and they were not instructed against it. We therefore audit what they actually did, in the disclosures below.

Before each matchup we freeze the provider and dated model identifier, system prompt, prompt template, sampling parameters, preview budget, retry policy, and the deterministic fallback used when an agent repeatedly emits an unusable action. Raw requests and responses are retained subject to provider terms. Each agent plays both throwing orders on the same evaluation roots and under the same execution-noise distribution as π_3, which plays at full deployed strength throughout. We report dScore, win score, separate win/draw/loss frequencies, invalid-output rate, latency, and token use where the provider exposes it. Consumer chat interfaces are not used for quantitative results because they do not expose a sufficiently pinned model and request configuration; agentic runs are reported with the dated model identifier and the harness that issued the calls.

Disclosures. Four asymmetries in this protocol are worth stating explicitly, because three of them favour the language model rather than π_3. First, the preview endpoint returns the champion's *own* value estimate for the resulting position, so a preview-enabled agent can query our policy's evaluation function; the plain text record does not contain it. Second, the API and agentic harnesses differ in how that estimate reaches the model. The driver used for the hosted open-weight agents injects π_3's post-shot evaluation into the model's conversation history on every turn, whereas the agentic players obtain it only by spending a preview. The zero-preview conditions therefore receive less information. Third, the preview budget is matched within the tool-enabled rows but not between the zero-preview rows and the open-weight rows. Fourth, the power play is not offered symmetrically. Both sides are limited to one per game, but a language-model agent may call it before the first throw of any end in which it holds hammer, on either wing, or decline it; π_3's is applied by a fixed online rule that fires only in the final end of the match, only when π_3 holds hammer in that end, and that it cannot decline, its value head choosing only the wing. The asymmetry costs π_3 in frequency rather than in judgement: its conditioning vector carries only within-end progress, alternation parity, and the side on turn, with no end index, running score, or ends remaining, so it has no representation over which a power-play timing decision could be expressed, and its per-end score shows no trend with end index over standard ends ($r = 0.04$, $n = 138$). In the matches reported here the agents

took the power play in 84% of matches and across seven different ends, whereas π_3 took it in 34%, always in the final end. This is why Table 3 separates standard from power-play ends. Side assignment is exactly balanced for the Claude Opus 5 cells (each run plays both throwing orders for a full match) and nearly so for GPT-6-Astra (36 to 38 ends); for the open-weight cells it is 11 to 8, because two matches were truncated by the provider's daily token cap. Truncation depends on cumulative token spend rather than on the position, so it is not outcome-selective, but neither is it a randomized stopping rule. One further roster entry, a wrapper model that shares a rate-limit bucket with an agent already in the table, was aborted before scoring and contributes no ends. Finally, the Claude zero-preview cell was extended from two runs to four after an interim look, to obtain power on the headline comparison; the extension was fixed before the additional runs were read and preserved side balance, but its nominal interval does not account for this interim decision. End-level standard errors in the table are descriptive: ends share match history, and agents adapt within runs. A nonsignificant score difference does not establish equivalent strength.

Information available to the agentic players. The Claude Opus 5 players ran inside an agent harness with shell access to the machine hosting the arena, whereas the replication harness described above withheld simulator source and other runs' records by construction. We therefore audited the full tool-call logs of all six Claude Opus 5 runs. Five of the six—every zero-preview run and one of the two tool-enabled runs—issued no filesystem commands at all: their knowledge of the physics came entirely from the arena's own responses, which return the solved intended action together with the resulting board, so a player accumulates roughly forty action–outcome pairs over a match. That is how these agents recovered the ballistics during play. One tool-enabled run is the exception: it listed the arena directory and read match records belonging to *other* runs. It read no source and no configuration files, but it did see other agents' logged transitions, which the replication protocol forbids. That run is one half of the tool-enabled cell and contributes no ends to the zero-preview result on which our main claim rests; within the tool-enabled cell the two runs differ by 2.08 ± 0.93 points per end in the direction of the better-informed run, a post-hoc comparison of thirteen ends each that we report rather than rely on.

Balanced power-play experiment. Table 4 uses 50 single-end matches against Claude Opus 5 in the zero-preview condition. Each match is created as a single end with a fixed hammer holder, so the power play is applied to π_3 by its usual rule when π_3 holds hammer and is called by the agent, on the wing of its choosing, when the agent does. The design fixed 25 matches per holder before any were played, with the agent's side alternating within each arm (13/12 in both). The agent was told that each match is a single end and that its only objective is the margin in that end. The run was interrupted three times by an account-level spending limit and resumed; eight matches abandoned mid-end by an interruption were discarded and replayed from scratch under the same label. Interruptions depended on cumulative spend rather than on the board, and no completed match was discarded. Interim results were inspected during these pauses (t ranged from 1.5 to 2.4), but the stopping rule was never changed. The agent played through several fresh contexts, each working a list of matches in order, and later contexts were assigned to a single arm, so within-context learning is not balanced across arms. Agents could read the arena's documentation, parts of its source, and logged match records, including other runs'. None ran the simulator locally, queried the solver or π_3's value head, or used a preview, which we verified from the full tool-call logs. This access can only have helped the agent. Excluding the one context that read the inverse solver and noise configuration in detail leaves the difference at $+1.13 \pm 0.58$.

I Additional Arena Result Details

Invalid-output rates for the API-driven agents were 18.9%, 22.4%, and 6.0% for `gpt-oss-120b`, `gpt-oss-20b`, and `qwen3.8-27b`, respectively.

The three hosted open-weight agents behaved as one group: pooled over 50 standard ends, π_3 outscored them by $+0.960 \pm 0.326$ points per end ($t = 2.95$) and won 64% of ends. Individually none reaches significance at this sample size, but the three estimates are tightly consistent. Hammer assignment is endogenous—it follows from who scored in the previous end—and ran against us in that group, so this margin understates π_3's advantage. The frontier agents are a different matter. Against GPT-6-Astra π_3 was ahead, $+0.443 \pm 0.298$ over 61 standard ends, though not significantly.

Claude Opus 5, the stronger of the two, played π_3 level over 54 standard ends with no preview access, $+0.056 \pm 0.320$, and across both preview conditions its result on standard ends is as close to exactly level as the data admit: 40 ends won and 40 lost. Granting it two inverse-solver previews per turn did not change this ($+0.038 \pm 0.501$ over 26 standard ends). The level result reflects two competent players rather than two policies failing in complementary ways: on standard ends π_3 scored $+1.069 \pm 0.424$ in the 29 it held hammer and conceded -1.120 ± 0.371 in the 25 it did not. Across every cell, all rejected turns were physical rather than syntactic, such as requesting a draw to an unreachable spot.

Power-play ends are where π_3 and Claude Opus 5 separate (Table 4; protocol in Appendix H). Both sides convert the power play, but π_3 converts it by more: $+2.320 \pm 0.386$ points per end against Claude Opus 5's $+1.120 \pm 0.371$, a difference of $+1.200 \pm 0.536$ (Welch $t = 2.24$, 95% CI $[0.15, 2.25]$; permutation $p = 0.035$). Equivalently, with the power play shared equally, π_3 outscored Claude Opus 5 by $+0.600 \pm 0.268$ points per end and won 30 of 50 ends. Weighting the two strata as they would occur with one power play per side in an eight-end match gives $+0.19 \pm 0.21$ points per end. Appendix J summarizes what the Claude Opus 5 players reported about their own play, including their recovery of the simulator's ballistics during a match.

$$\begin{array}{cccccc} s, c & \xrightarrow{E_\theta} & h & \xrightarrow{G(\cdot,a)} & \hat{h}' & \\ s, c, a & \xrightarrow{\text{Sim}} & s', c' & \xrightarrow{\text{sg } E_{\bar{\theta}}} & h'_{\text{target}} & \end{array} \qquad \mathcal{L}_{\text{cons}} = d(\hat{h}', h'_{\text{target}})$$

Figure 6: Latent-consistency training. The online branch predicts the next representation from an action; the target branch encodes the simulator's actual next board and receives no gradient.

J What the Claude Opus 5 Players Reported

Each Claude Opus 5 player wrote a short account of its own match, which we summarise because it says something about what the comparison measures. Three themes recur across runs. First, *system identification*: given only the logged transitions it could see, the agent recovered the simulator's ballistics during play (travel distance as a function of release speed, the resting position that follows from it, and a back-loaded curl profile) and then largely abandoned the solver-backed intents in favour of raw (v, θ, ω, y_0) parameters. Part of what the strongest cell measures is therefore the ability to identify a simulator from logged play, which is not available on real ice. GPT-6-Astra, which received the same responses to its own throws, never did this: it drew on 79% of its throws and issued no raw-parameter shots, against 55% and a quarter for Claude Opus 5. Second, the agents converged on the same read of our policy's strength: its hit-and-stick, a takeout that leaves the shooter in scoring position, was repeatedly described as close to automatic, and one agent concluded that forcing π_3 to hit is not a defence. Third, on the tool-enabled condition the agents found the *deterministic* fields of a preview (feasibility, removal reliability, whether the shooter survives) far more useful than the single simulated realization, which they reported as actively misleading because previews finish nearer their target than committed draws do. One agent also reported that our champion's own value estimate, which a preview exposes, occasionally ranked positions it considered clearly dominated above better ones; this is an adversarial, self-selected anecdote from an interested party rather than a measurement.

K Deployment-Time Search and Representation Ablations

Both arms use the same checkpoint and differ only in how a shot is chosen. The search arm is the deployed selector: 48 proposals from the policy head, each simulated under eight execution-noise realizations and ranked by mean value-head score. The no-search arm draws one shot from the same policy head at the same sampling temperature and throws it. Roots are drawn from the human validation partition at every horizon $h = 1, \ldots, 10$, each played in both throwing orders under the same execution noise, giving 40 cells and 400 scored ends. The search arm scores $+1.2425 \pm 0.1059$ dScore per end (standard error over ends; 0.0899 when clustered by root), $t = 11.7$, and wins 71.8% of ends. Per-horizon values rise from $+0.50$, $+0.13$, and $+0.78$ at $h = 1, 2, 3$ to $+1.95$ at $h = 9, 10$, and the two throwing orders give $+0.97 \pm 0.15$ and $+1.52 \pm 0.15$. An independent one-ply check on an earlier checkpoint, using a stored chance-correct expectimax reference at $h = 2$, gives the same

sign with a per-decision advantage of $+0.107 \pm 0.048$ over a uniform policy sample, consistent in magnitude with a per-end gap accumulated over ten stones.

To isolate the auxiliary objectives, we retrain the final response with each loss weight set to zero in turn, using the same initialization, replay corpus, architecture, optimization budget, augmentation, and checkpoint rule as the full model. Both retrained models select the same epoch and run the same number of steps as the full model, and all three are scored on the same held-out shards under the same loss configuration. The full model obtains policy-target NLL 5.2546 and value MSE 3.683, compared with 5.2650 and 3.688 without latent consistency, and 5.2546 and 3.684 without the value-NLL term. The value-NLL ablation is the smaller intervention of the two, a relative L_2 parameter distance of 0.18% from the full model against 1.66% for the consistency ablation, which follows from its mechanism: the log-variance head receives gradient only from that term and is never read when ranking candidates, so zeroing it freezes the head and leaves only an indirect path through the mean head and the shared trunk. Because these experiments retrain only the final response, they measure the marginal contribution of each objective at that stage rather than its cumulative effect across the full response ladder; the warm start they share was itself trained with both objectives active.

L Value Calibration Protocol

Calibration is measured on the human competition *test* partition of the holdout split, which is used for neither training nor checkpoint selection, giving $n = 3{,}851$ pre-shot states with their realized end outcomes. For each state the value head emits a Gaussian (μ, σ^2) over the signed end margin from the team-to-move perspective. We report mean squared error $\mathbb{E}[(\mu - y)^2]$, the standard Gaussian negative log-likelihood $\frac{1}{2}\mathbb{E}[\log(2\pi\sigma^2) + (y-\mu)^2/\sigma^2]$ (note that the training logs record a constant-free variant, which is 0.99 here), and central-interval coverage $\Pr(|y - \mu| \le z_q \sigma)$ at $q \in \{50, 80, 90\}\%$. The coverage confidence interval is a 2,000-resample nonparametric bootstrap over states.

M Expert Evaluation Protocol

Our collaboration with U.S. Olympic Curling Team staff informs arena controls, scenario selection, and presentation of policy and value outputs. The formal study uses 30 positions drawn from a competition-level human holdout (three states at each of ten horizons). For each recorded pre-shot state, we pair the recorded human action, inverse-recovered from the adjacent boards, with π_3's action on that identical state. Both candidates are rendered through the same interface, without source labels, and their left–right order is randomized independently across trials.

Instrument as administered. The design registered two forced-choice questions per pair—"Which action is more human-like?" and "Which action is more strategic?"—each admitting an "indistinguishable" response. The round reported here departs from that design in three respects, and we report it as administered rather than as planned. First, only one question was put to the reviewers: *which shot looks more optimal?* We therefore report no human-likeness result at all; that outcome is simply not measured here. Second, the response was a forced binary choice, so no ties were recorded and the tie-handling rule in the aggregation plan is inert. Third, two reviewers each returned a completed 30-position sheet, yielding 60 judgments over the same 30 positions. We therefore report the pooled selection rate over all 60 judgments together with an interval clustered by position, since the two sheets are not independent samples of positions; the reviewer and position identifiers are retained in the released evaluation records. The reviewers agreed on 20 of the 30 positions ($\kappa = 0.32$), and their individual selection rates were 40.0% and 46.7%. One reviewer showed a mild preference for the left-hand option (choosing it 19 of 30 times), the other none (15 of 30); because the answer key places π_3 on each side exactly 15 times, a pure position preference cancels out of the pooled estimate.

What the reviewer sees. Each option displays the shared pre-shot board and the post-shot board that the action produces, with the delivery drawn as a dotted path. The human option's post-shot board is the recorded one; π_3's is a single separately seeded execution-noise realization of its chosen action. Execution quality is therefore visible to the reviewer, and an unlucky realization can make

a strategically sound choice by π_3 look poor. This is a limitation of the present instrument rather than a property of the policy: a cleaner version would either display both options' intended resting positions or average π_3's displayed board over realizations. Reviewers do not see the realized end score. The instrument records both the intended and the displayed parameters for π_3, so the size of each execution-noise draw is recoverable after the fact. Expressed in units of the realized per-dimension noise standard deviation and combined as a Euclidean norm, draws on trials where the reviewers rejected π_3 were larger than on trials where it was chosen (1.88 against 1.63 over the 60 judgments; Spearman $\rho = -0.17$, $p = 0.19$, or $\rho = -0.21$ clustering by position). This is not significant and it is a post-hoc test, but its sign is the one predicted if the displayed realization leaks execution luck into a judgement intended to be about shot selection.

N FREE-GUARD-ZONE LEGALITY

With horizon h = throws remaining including the current throw and shots_in_end $= 10$, the thrown-stone index is $k = 11 - h$. The no-takeout rule is active for $k \in \{1, 2, 3\}$ (i.e. $h \geq 8$). A post-state is illegal iff some opponent rock that was in play before the throw is out of play after it; movement that keeps it in play and removal of one's own rock are legal. The protected set is all in-play opponent rocks, including the pre-placed one. Illegal posts are replaced by the pre-shot board (throw forfeited, displaced rocks restored). The operator is applied to every simulated candidate post-state in collection, search, and evaluation.