REASONING ON THE SIMPLEX: GEOMETRIC FIXED-POINT MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped reasoners spend test-time compute by iterating a weight-tied map, but a small residual does not mean the state is a fixed point when that map lives in unconstrained latent space. We propose Geometric Fixed-Point Reasoning (GFPR), in which the iterated state is the prediction itself: a field of categorical beliefs on a product of simplices, whose $\arg\max$ is the answer at every step. Because the state is a belief, task structure can be imposed through compact convex relaxations, either as structured readouts or directly in the recurrent state; in the latter case the update remains a continuous self-map, so a fixed point exists for any parameters. At about 7M parameters, GFPR reaches 95.1% exact match on Sudoku-Extreme, 92.0% on Maze-Hard, and 100% sequence accuracy on S_5 length 128, above the published FPRM numbers at the same scale. The same update also trains a 201M language model on FineWeb-Edu in which each site is a distribution over the vocabulary; with 24 Picard steps it is above GPT-2 small on four zero-shot multiple-choice tasks and above GPT-2 medium on ARC-Easy.

1 INTRODUCTION

Reasoning benchmarks often mix easy and hard instances, and the hard ones need more internal computation to solve reliably. A popular response is to add test-time compute without training a larger network, so hard examples can receive more inference steps (Snell et al., 2024). In large language models, one common mechanism is to generate a longer chain of thought before the final answer (Wei et al., 2022; Kojima et al., 2022). That works well on many open-ended math and logic prompts, but chain-of-thought reasoning is not always the right interface: it ties computation to autoregressive text, grows latency with every token, and offers no grid-shaped state on which to enforce local constraints (Wang et al., 2025).

A different way to spend test-time compute is to apply the same neural block again and again: keep a latent state and update it with a weight-tied map, often for far more steps at inference than during training (Jolicoeur-Martineau, 2025; Wang et al., 2025; Movahedi et al., 2026; Huang et al., 2026). Read this way, looped reasoning looks like fixed-point iteration: repeat an update until successive states agree, then decode an answer from the settled state (Bai et al., 2019; Movahedi et al., 2026; Huang et al., 2026). FPRM (Movahedi et al., 2026) makes this explicit and stops when the iterate residual is small, unlike TRM/HRM-style ACT halting (Graves, 2016; Jolicoeur-Martineau, 2025; Wang et al., 2025).

Halting on a small residual only certifies an answer if the looped map has a fixed point that the iteration actually reaches. For an arbitrary latent vector in $\mathbb{R}^d$ the standard guarantee is a contractive map. Movahedi et al. (2026) show that small residual scaling is sufficient for contraction, note that it is not guaranteed in practice, and add a decaying damping step to suppress the resulting oscillations. On their public Sudoku checkpoint, however, the state at which inference halts is not a fixed point, and even long after the halt the one-step map is typically not locally contractive (Section 2).

We propose Geometric Fixed-Point Reasoning (GFPR), which keeps damped fixed-point iteration but changes the space in which the iterate lives. The state is a field of categorical beliefs, one probability vector per grid cell or token, and every update returns to this product of simplices through a softmax readout. After damped Picard iteration the answer is the $\arg\max$ of those beliefs, with known outputs pinned throughout, and the state can be read at any step (Section 3). The same beliefs also support task-specific convex structure, used either as a structured readout or directly

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.

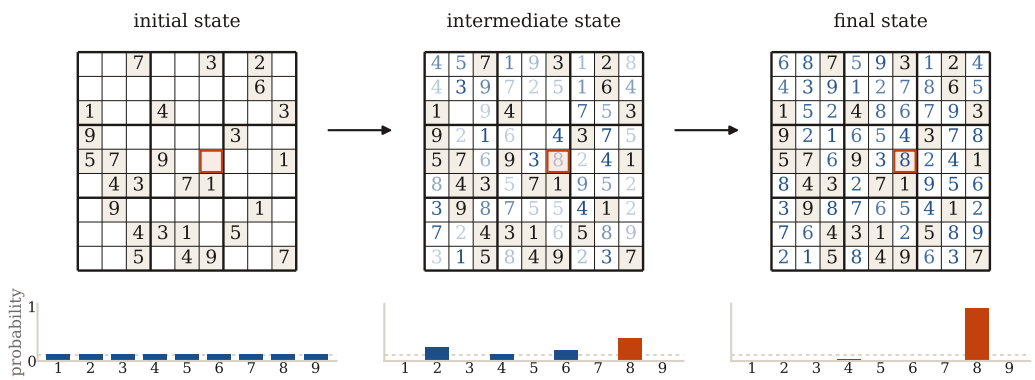

Figure 1: In GFPR the state is a field of simplex-valued beliefs: at each site, a distribution over local outputs (e.g. Sudoku digits or vocabulary tokens) together with auxiliary register coordinates that store internal state but are not decoded as answers. At test time we unroll damped Picard steps; the prediction is $\arg\max$ over the output block of the final belief, with known outputs pinned throughout.

as the recurrent state (Section 3.4). When the recurrent state itself is constrained to a compact convex set, the update remains a continuous self-map, so a fixed point exists; which one the iteration reaches is decided by training (Section 3.2). At ∼7M parameters, GFPR reaches 95.1% exact match on Sudoku-Extreme and 92.0% on Maze-Hard, compared with 94.2% and 87.0% for FPRM at the same scale (Section 4). The same update, trained as a 201M language model on FineWeb-Edu, is compared with GPT-2 on standard zero-shot multiple-choice tasks (section 4.4).

To sum up, our contributions are:

- We introduce GFPR, a looped reasoner whose state is a field of categorical beliefs and whose answer is the $\arg\max$ of that state (section 3).
- We show that this state lets us impose structural constraints for classical reasoning tasks such as Sudoku, mazes, and permutations (section 3.4).
- We evaluate GFPR on Sudoku-Extreme, Maze-Hard, and S_5 state tracking, where it is above the published FPRM numbers at similar scale (section 4).
- We train a 201M simplex language model on FineWeb-Edu and compare it with GPT-2 small and medium (section 4.4).
- We study the iteration dynamics of GFPR and compare them with other looped reasoners, showing that GFPR settles to a fixed point while FPRM typically halts at a state that is not one (section 2).

2 Background: Looped Reasoning as Fixed-Point Iteration

A looped reasoner applies one weight-tied map $F_\theta(\cdot\,; x)$ to a state u. With damping $\beta \in (0, 1]$,

$$u_{k+1} = T_\beta(u_k), \qquad T_\beta(u) = (1-\beta)\,u + \beta\,F_\theta(u; x), \tag{1}$$

whose fixed points satisfy $u^\star = F_\theta(u^\star; x)$. FPRM (Movahedi et al., 2026) halts on a small relative iterate residual with decaying step size; TRM and HRM use ACT or a step cap (Graves, 2016; Wang et al., 2025; Jolicoeur-Martineau, 2025); DEQs solve for $u^\star$ implicitly (Bai et al., 2019).

Existence and reachability. Existence of a fixed point and convergence of equation 1 are different questions. Contraction answers both: if F_θ is L-Lipschitz with $L < 1$, Banach's theorem (Banach, 1922; Ortega & Rheinboldt, 2000; Kelley, 1995) gives a unique fixed point and global convergence, and damping preserves $1 - \beta + \beta L < 1$. For a trained network one usually checks local stability at a putative fixed point $u^\star$ through the Jacobian J_F: eigenvalues λ of J_F become $1 - \beta + \beta\lambda$ under damping, and $u^\star$ attracts nearby states when all lie in the unit disk. If $\rho(J_F) < 1$ this holds for every β; if $\rho(J_F) > 1$, F_θ is not locally contractive in any norm (Horn & Johnson, 2012), yet damped

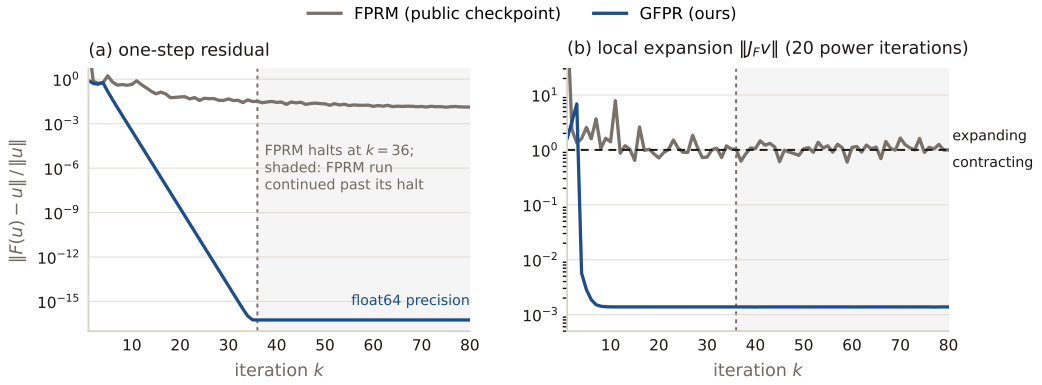

Figure 2: One randomly chosen Sudoku-Extreme test puzzle that both FPRM and GFPR solve exactly. The dotted line marks where FPRM's own rule halts ($k = 36$); in the shaded region we keep running FPRM to see whether it settles. **(a)** One-step residual $\|F(u) - u\|/\|u\|$ in float64. GFPR's residual falls to float64 precision; FPRM's stays near 10^{-2} before and after its halt. **(b)** Local amplification probe at each step: K_{pi}=20 power iterations on JVPs (float32), plotted as $\|J_F v\|$ after renormalization (not a certified $\rho(J_F)$ when J_F is non-normal; section G). The dashed line marks unit amplification. On the final states of this puzzle only, restarted Arnoldi gives largest Ritz moduli $\approx 1.3 \times 10^{-3}$ (GFPR) and ≈ 1.04 (FPRM).

iteration may still converge. Non-normal J_F can make the residual non-monotone, and a small halt residual need not place the state near $u^\star$ when J_F has an eigenvalue close to one (Trefethen & Embree, 2005; Kerg et al., 2019).

Brouwer's theorem (Brouwer, 1911) guarantees a fixed point for every continuous self-map of a nonempty compact convex set, with no contraction and for every θ, but not uniqueness or convergence. GFPR's belief polytope is of this kind (section 3); training must make the correct equilibrium reachable from the uniform start.

Are FPRM's halted states fixed points? FPRM iterates a free latent $y \in \mathbb{R}^d$ and uses pre-norm blocks, residual scaling, and decaying step size s to encourage contraction (Movahedi et al., 2026). We run its public Sudoku checkpoint with the authors' loop (relative residual threshold 0.1, $s \leftarrow 0.997s$ on stall; supplementary material).

On 1000 held-out Sudoku-Extreme puzzles, 89% are solved, yet no halted state has relative RMS residual below 10^{-3} (median 3.1×10^{-2}). With halting disabled for 8000 steps, the same probe with K_{pi}=10 still reports $\|J_F v\| > 1$ on 75% of 128 held-out puzzles (supplementary material); we have not cross-checked that fraction with Arnoldi, so it is a diagnostic of local expansion under the probe rather than a population claim that $\rho(J_F) > 1$.

Figure 2 is a single-puzzle case study both models solve. GFPR's residual reaches float64 precision (Arnoldi largest Ritz modulus $\approx 1.3 \times 10^{-3}$ on this puzzle); FPRM's stays near 10^{-2} through its halt at $k = 36$. Near the halt the dominant Jacobian mode is the pair $\lambda \approx -0.13 \pm 1.03i$ ($|\lambda| \approx 1.04$). Any step $s < 0.966$ would damp it, but the halt fires at $s \approx 0.99$, where $|1 - s + s\lambda| \approx 1.03$: the update FPRM applies at the state it returns still expands along this mode.

3 Method

3.1 Belief state and update

GFPR represents a problem with n sites (grid cells or tokens) by a state $p = (p_1, \ldots, p_n)$, where each p_i is a probability vector over K output symbols and A auxiliary coordinates. The output block is what the task decodes, for example digits in Sudoku; the auxiliary coordinates act as registers for intermediate computation and are never read out. Let $\mathcal{K}$ be the set of sites whose output is given by the input (for example Sudoku clues), and let $\mathcal{P}_x$ be the product of simplices in which every site of

$\mathcal{K}$ is fixed to the one-hot vector of its given symbol. With damping $\beta \in (0, 1]$, one step is

$$F_\theta(p;\, x) = \Pi_x\Big(\mathrm{softmax}\big(f_\theta(W_{\mathrm{in}}p + \mathrm{Enc}(x))\big)\Big), \qquad T_\beta(p) = (1-\beta)\, p + \beta\, F_\theta(p;\, x), \tag{2}$$

where f_θ is a Transformer over all sites, the softmax acts per site, and Π_x overwrites every site of $\mathcal{K}$ with its one-hot vector. The prediction at site i is the $\arg\max$ over the output block of the settled p_i. Figure 1 illustrates this settling process on a Sudoku puzzle.

Since $\mathcal{P}_x$ is nonempty, compact, and convex and T_β maps it continuously into itself, a fixed point exists for every θ, x, and $\beta \in (0, 1]$ (section 2).

Why a simplex. Existence alone does not single out the simplex: any continuous map of a compact convex set into itself has a fixed point, so a bounded latent $z \in [-1, 1]^d$ updated by $\tanh(f_\theta(z; x))$ has one too. What the simplex adds is that every intermediate state is itself a prediction. First, the iteration can be read at any step: the $\arg\max$ of p_i is the current answer at site i and p_i is the model's confidence in it, with no readout head in between. Second, the residual measures how much the prediction moves. For any two distributions $|a_j - b_j| \le \mathrm{TV}(a, b)$, so a site whose top two output probabilities differ by more than twice its per-step TV keeps its $\arg\max$ on that step; in a latent chart, a small step in z bounds the change of the decoded answer only through the Lipschitz constant of the head. Third, the correct solution is a known point of the state space, a vertex of $\mathcal{P}_x$, so the loss pulls the state toward it directly and the distance to it can be tracked along the trajectory. Fourth, task constraints are statements about beliefs, so they can be imposed on intermediate states rather than checked after decoding (section 3.4).

3.2 Training and inference

Training unrolls the iteration from the uniform state, or with probability 0.25 from a random Dirichlet state, in two phases. A first rollout of at most D steps runs without gradients and stops early once every puzzle in the batch has a per-site TV step below a tolerance for two consecutive steps. From its end state $p^{(0)}$, a tail of m steps $p^{(j+1)} = T_\beta(p^{(j)})$ is differentiated. With $\tilde{p}_i$ the renormalized output block of site i at the final state $p^{(m)}$ and $\bar{\mathcal{K}}$ the free sites, the loss is

$$\mathcal{L} = \frac{1}{|\bar{\mathcal{K}}|} \sum_{i \in \bar{\mathcal{K}}} -\log \tilde{p}_{i,y_i} + \lambda_{\mathrm{aux}}\, a(p^{(m)}) + \lambda_{\mathrm{res}}\, \frac{1}{m} \sum_{j=0}^{m-1} \frac{1}{n} \sum_{i=1}^{n} \big\| \hat{F}_{\theta,i}(p^{(j)}; x) - p_i^{(j)} \big\|_2^2, \tag{3}$$

where $a(p)$ is the mean auxiliary mass on free sites and $\hat{F}_\theta = \mathrm{softmax}\big(f_\theta(W_{\mathrm{in}}p + \mathrm{Enc}(x))\big)$ is F_θ before Π_x. The last term is the one-step residual along the tail; because it is taken before pinning, it also asks the network to reproduce the given sites. We use $\lambda_{\mathrm{res}} = 0.02$ on Sudoku and $\lambda_{\mathrm{res}} = 0$ on Maze; depth distributions and other per-task settings are listed in the supplementary material.

Which fixed point is learned. Existence does not say which fixed point the test-time loop reaches from the uniform start (section 2). Training aligns F_θ with that trajectory: the cross-entropy is taken at the end of a rollout from the same start, so gradients act on the states that inference actually visits rather than on equilibria it may never approach. The Dirichlet starts ask that the labeled vertex also be reached from a spread of initial beliefs, which widens its basin.

On Sudoku the rollout cap D is redrawn every batch from a heavy-tailed distribution (Maze uses a fixed D), so the same weights must give the right answer after short and long rollouts. A map that passes through the solution at one depth and then moves on is penalized at the others, so the answer has to persist. Short draws also end the rollout before the state settles, so the loss sees intermediate states as well; training only at settled states would leave the map free to form confident wrong attractors in regions the loss never visits.

With $\lambda_{\mathrm{res}} > 0$ on Sudoku, the residual term additionally asks the state the model settles on to be stationary, not merely correct at the end of the tail. On Maze, $\lambda_{\mathrm{res}} = 0$ and only the final answer is supervised. Near a fixed point, where the Jacobian varies slowly along the tail, backpropagation through the m tail steps approaches a truncated Neumann approximation of the implicit fixed-point gradient, which involves $(I - J_F)^{-1}$ (Bai et al., 2019).

None of this certifies convergence at test time. Training rollouts are tens of steps long (mean $D \approx 32$ on Sudoku), while test budgets reach $35{,}000$ steps, so behavior beyond the training horizon is extrapolation; a few held-out Sudoku puzzles settle to a stable but incorrect grid.

At test time GFPR starts from the uniform state with the given sites pinned, applies $p_{k+1} = T_\beta(p_k)$ for K steps (on S_5, until the total-variation step falls below a tolerance; budgets in the supplementary material), and reads $\arg\max$ on the output block of p_K. No labels, verifier, or restarts are used at test time.

3.3 Language models: wavefront decode

The same state applies to a token sequence. Site t holds a belief p_t over the vocabulary, and training fits the output block of p_t to the next token x_{t+1}, with the embedding of the true x_t as conditioning. One Picard step is one causal pass of f_θ. On Sudoku and mazes every free site is updated for the full inference depth and the whole answer is read at the end (section 3.2). Under causal attention the update at site t depends only on sites $\le t$, so p_t is a prediction of x_{t+1} only after $x_1, \dots, x_t$ are fixed. Open-ended generation therefore alternates Picard steps with commits, and we use *wavefront* decode for it. Wavefront decode is purely an inference procedure: the loss equation 3 is teacher-forced, so F_θ is never trained on an expected embedding. The multiple-choice scores of section 4.4 need no generation and are computed from teacher-forced beliefs.

Wavefront decode writes the sequence left to right. It tracks a frontier f: the position of the next token that is not yet fixed. When the belief p_{f-1} at the site before the frontier looks ready, we commit $x_f = \arg\max p_{f-1}$, pin that site to a one-hot vector, and move f forward by one. While we wait for readiness, we do not freeze the entire suffix: up to W sites ahead of f (the soft-ahead width) keep taking Picard steps so later beliefs can start to form before the prefix is complete.

Embeddings follow the same split. Sites that are already committed feed the network their true token vectors. Inside the window, site i conditions on a soft embedding built from the belief at $i-1$,

$$e_i = \sum_w q_{i-1,w}\,\mathrm{Emb}(w), \qquad q_{i-1} = \mathrm{softmax}(\log p_{i-1}/\tau), \tag{4}$$

where q_{i-1} is a temperature-sharpened version of p_{i-1}. Sites beyond the window are not updated until the frontier moves and they enter the window. On the very first segment, the given prompt is run with teacher-forced embeddings only, so the readiness test at f is not applied to beliefs that are still essentially uniform from the cold start.

Readiness of p_{f-1} is a gap between its top two probabilities, a total-variation residual that has stayed small for several steps, or the per-token step cap. A minimum number of steps is required after every commit. The first steps after a commit are small even when the state is far from equilibrium, and a raw residual test treats that pause as convergence. Pinning p_{f-1} to the committed one-hot makes the soft embedding at the next site match the ordinary token embedding of x_f.

Because the update at site t ignores sites $> t$, prefix order is the order the dependencies allow. The soft window lets up to W later beliefs move before their prefix is frozen. A margin test can commit a sharp belief that is still moving; a residual test can wait out a belief that has already converged to a flat distribution. Default W, tolerances, and the implementation are in section E.

3.4 Task-specific convex constraints

The product-of-simplices state does not couple sites. We encode global task structure by a compact convex relaxation $\mathcal{C}(x)$: a terminal structured readout for Sudoku and Maze-Hard, and the recurrent state for S_5. Let q denote the structured answer variable, distinct from the state p of section 3.1. For nonempty compact convex $\mathcal{C}(x) \subset \mathbb{R}^d$ and a continuous convex regularizer Ω, define

$$\begin{aligned} \Phi_x(z) &= \max_{q \in \mathcal{C}(x)} \{\langle z, q\rangle - \Omega(q)\}, \\ \Psi_{\mathcal{C}(x)}(z) &= \arg\max_{q \in \mathcal{C}(x)} \{\langle z, q\rangle - \Omega(q)\}. \end{aligned} \tag{5}$$

Here $\Psi_{\mathcal{C}(x)}$ is a structured analogue of softmax: it maps unconstrained scores to an answer state satisfying the convex task constraints. If the maximizer is unique, compactness and continuity make $\Psi_{\mathcal{C}(x)}$ continuous in z.

When the structured variable q is used as the recurrent state, we update it as

$$T_{\beta,\mathcal{C}}(q;x) = (1-\beta)q + \beta\,\Psi_{\mathcal{C}(x)}\big(f_\theta(q,x)\big), \qquad \beta \in (0,1]. \tag{6}$$

Both terms lie in $\mathcal{C}(x)$, so convexity keeps every update in $\mathcal{C}(x)$. Thus $T_{\beta,\mathcal{C}}$ is a continuous self-map, and the fixed-point existence argument of section 2 applies.

Instantiations. For Sudoku, q_{rcd} is the belief that cell (r,c) takes digit d. The relaxation $\mathcal{C}_{\mathrm{Sud}}(x)$ requires a distribution on each cell, unit total mass of each digit in every row, column, and 3×3 box, and $q_{rc,d_{rc}} = 1$ on clues (equation 9). Its points may be fractional, so membership does not itself certify a valid grid. We use the power regularizer $\Omega_\alpha(q) = \frac{1}{\alpha(\alpha-1)}\sum_i q_i^\alpha$ with $\alpha > 1$ and evaluate $\Psi_{\mathcal{C}_{\mathrm{Sud}}(x)}$ through its dual (section D).

For Maze-Hard, parent pointers induce a unit flow from the goal to the start on directed grid edges. With incidence matrix B_x and right-hand side b_x (source at the goal, sink at the start),

$$\mathcal{C}_{\mathrm{flow}}(x) = \left\{ f \in [0,1]^{|E_x|} : B_x f = b_x \right\}. \tag{7}$$

Flow conservation does not enforce a simple or shortest path; we decode the resulting parent field and evaluate the route separately.

For S_5, the structured state is a doubly stochastic matrix $X \in \mathcal{B}_5$, parameterized as $X = \sum_{\pi\in S_5} w_\pi P_\pi$ over the 120 permutation matrices P_π. The weights w_π are obtained by applying either softmax or α-entmax (Peters et al., 2019) to the permutation scores.

Fenchel–Young readout. For a target $y \in \mathcal{C}(x)$ we use the Fenchel–Young loss (Blondel et al., 2020)

$$L_{\Omega,\mathcal{C}}(z,y) = \Phi_x(z) + \Omega(y) - \langle z, y\rangle; \tag{8}$$

when the maximizer in equation 5 is unique, $\nabla_z L_{\Omega,\mathcal{C}}(z,y) = \Psi_{\mathcal{C}(x)}(z) - y$. In Table 1, Sudoku and Maze retain the original recurrent state p, including its auxiliary coordinates, and apply $\Psi_{\mathcal{C}(x)}$ only to the final scores. Thus the structured solver is a terminal readout rather than part of the recurrent loop. The constrained S_5 model instead uses the doubly stochastic matrix X as its recurrent state and applies the structured update at every step. Solver details and the entmax α ablation are in section D.

4 Experiments

4.1 Tasks

We evaluate on three structured benchmarks used by recent looped reasoners (Wang et al., 2025; Jolicoeur-Martineau, 2025; Movahedi et al., 2026), all scored by exact match of the full answer, and on zero-shot language-model multiple choice.

Sudoku-Extreme. 9×9 puzzles with unique solutions, selected to be hard (Wang et al., 2025). Training uses the 1,000-puzzle split with validity-preserving augmentations (digit relabeling, band and stack permutations, transposition). Checkpoints are chosen on a stratified 1,000-puzzle validation slice of the `test_hard` file in `sapientinc/sudoku-extreme-1k`; the headline number is on the separate public file `sapientinc/sudoku-extreme test.csv` (422,786 puzzles) used by HRM and FPRM, which is not used for model selection (section C). Headline evaluation matches the released FPRM Sudoku-Extreme setup (Movahedi et al., 2026): one trajectory from the uniform state, damping $\beta = 0.7$, and at most 35,000 damped Picard steps (the same step cap as FPRM on this benchmark). Each cell is a site with $K = 9$ digits, and the clues are pinned.

Maze-Hard. 30×30 mazes whose shortest path is longer than 110 steps, with 1,000 training and 1,000 test mazes (Wang et al., 2025). Rather than predicting the route mask, GFPR predicts at every open cell the direction to its parent in a breadth-first tree rooted at the start; the route is recovered by following parent pointers from the goal and compared with the labeled route. We train with the eight dihedral symmetries of the square, which the parent field respects because it is recomputed from the transformed maze. We evaluate on all 1,000 test mazes, as in prior work on this benchmark.

Table 1: Exact-match accuracy (%). "–" was not reported. Quoted baselines are from Movahedi et al. (2026), except TRM-MLP (Jolicoeur-Martineau, 2025). †One trajectory. ‡A sample of 23,680 Sudoku test puzzles. Slash-separated parameter counts correspond to Sudoku-Extreme, Maze-Hard, and S_5, in that order.

Model	Params	Sudoku-Extreme	Maze-Hard	S_5
HRM	27M	55.0	74.5	–
TRM (attention)	7M	74.7	85.3	39.4
TRM (MLP)	5M	87.4	–	–
TRM + causal conv	7M	–	–	97.2
EqR	7M	93.0†	–	–
FPRM	7M	94.2	87.0	98.8
GFPR (ours)	7M / 7M / 5.6M	**95.1**†	**92.0**	**100.0**
GFPR with constraints (ours)	6.4M / 6.8M / 1.2M	**95.0**†‡	**96.0**	**100.0**

S_5 state tracking. Each instance is an initial arrangement of five elements followed by a sequence of permutations from the symmetric group S_5, and the model must output the final arrangement (Merrill et al., 2024). Following Movahedi et al. (2026), models are trained on sequences of up to 32 updates and evaluated on up to 128, which tests length generalization. The reported quantity is exact accuracy of the final arrangement. We enumerate the 120 permutations in lexicographic order and represent each input permutation and each target prefix product by its index. GFPR encodes every step as a site on the product of simplices over these group elements, with no coupling across sites, and scores a sequence correct only when every prefix is correct. The constrained variant keeps a doubly stochastic state $X = \sum_\pi w_\pi P_\pi$, where P_π are the 120 permutation matrices and the vertex weights w are a softmax of the scores $\langle Z, P_\pi \rangle$; the entmax variant replaces this softmax by α-entmax.

Language modeling. The same product-of-simplices state is trained as a causal language model on 6B tokens of FineWeb-Edu (Penedo et al., 2024) (section 3.3). The released run has 201M parameters (width 2048, 16 heads, two weight-tied blocks, vocabulary 16,384 plus 16 auxiliary coordinates, context 512). We score four zero-shot multiple-choice tasks with next-token log-likelihood, not generated text: ARC-Easy (Clark et al., 2018), SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2020), and HellaSwag (Zellers et al., 2019). GPT-2 small (124M) and GPT-2 medium (355M) (Radford et al., 2019) are the comparison points.

4.2 Baselines

We compare with looped reasoners at a similar parameter scale: HRM (Wang et al., 2025), TRM (Jolicoeur-Martineau, 2025), EqR (Huang et al., 2026), and FPRM (Movahedi et al., 2026). For Sudoku-Extreme and Maze-Hard, every model in Table 1 is reported on the same public test files: `sapientinc/sudoku-extreme test.csv` (N=422,786) and `sapientinc/maze-30x30-hard-1k test.csv` (N=1,000), as in HRM/TRM/FPRM (Wang et al., 2025; Jolicoeur-Martineau, 2025; Movahedi et al., 2026). We do not re-run the baselines; quoted accuracies come from Table 1 of Movahedi et al. (2026), with TRM-MLP from Jolicoeur-Martineau (2025). Fair comparison therefore requires matching not only the test puzzles but also the inference budget (unroll steps and damping; for FPRM, also residual tolerance and step-size decay). On Maze and S_5 our released runs use 1,024 Picard steps at $\beta = 0.3$ and up to 256 at $\beta = 0.7$, respectively (full protocol in section C). The training recipes also differ (for example, GFPR uses dihedral augmentation on Maze-Hard and FPRM does not), so Table 1 compares published systems and does not by itself isolate the effect of the state space. Unless noted, every method decodes a single trajectory.

4.3 Results

Table 1 summarizes the results. On Sudoku-Extreme, GFPR solves 95.1% of the 422,786 test puzzles (96.9% of cells) from a single trajectory, above FPRM (94.2%) and the other single-trajectory models at 5–7M parameters. EqR reaches 99.8% when it runs 128 restarts per puzzle and selects

Table 2: Zero-shot multiple-choice accuracy (%). ARC-Easy, SciQ, and PIQA are accuracy; HellaSwag is length-normalized accuracy. GFPR-LM is the 201M FineWeb-Edu run (EMA weights, $\beta = 0.5$, 6B tokens), scored by next-token log-likelihood after K damped Picard steps from the uniform state. GPT-2 small (124M) and medium (355M) use one forward pass in the same harness. All GFPR-LM columns are official full-split figures for this checkpoint (sweep through K=32 in section F); K=24 is the training depth, not a test-set choice. Bold marks the best of GPT-2 and K=24; the K=8 and K=16 columns show how the scores depend on depth.

			GFPR-LM, Picard steps K			
Task	GPT-2 S	GPT-2 M	1	8	16	24
ARC-Easy	44.2	49.2	32.0	28.4	49.5	**51.9**
SciQ	72.5	**75.3**	37.4	32.9	76.1	74.4
PIQA	62.4	**66.9**	55.6	53.8	62.8	63.1
HellaSwag	31.2	**39.5**	27.8	26.0	32.2	33.2

by residual (Huang et al., 2026); this breadth scaling is orthogonal to GFPR and could be combined with it. On Maze-Hard, GFPR solves 92.0% of the 1,000 test mazes, compared with 87.0% for FPRM. FPRM trains on Maze-Hard without augmentation, whereas GFPR uses dihedral augmentation, as TRM does. On S_5, trained on sequences of at most 32 updates and tested at length 128 with 320 effective layers, FPRM reaches 98.8% and plain TRM 39.4%; a causal convolution, which is not part of the original TRM, raises TRM to 97.2% (means over seeds, ± 0.9, ± 1.9, ± 2.5; Movahedi et al., 2026). GFPR, a 5.6M product-of-simplices model and a single seed, reaches 100.0% sequence accuracy on the 1,000 test sequences, which implies final-state accuracy. The constrained models of section 3.4 help most on Maze-Hard, where the unit-flow polytope raises exact match from 92.0% to 96.0%. On S_5 both the softmax and the α-entmax variants reach 100.0%. On Sudoku, with $\alpha = 1.75$, the consistency polytope scores 94.99% against 94.95% for plain GFPR on the same 23,680 random test puzzles, so it matches but does not yet improve on the product of simplices.

What the belief state shows. Because the state is the prediction, its failures can be inspected without a decoder. On Maze-Hard, 99.67% of cells carry the correct parent pointer although only 92.0% of mazes are solved exactly, and 99.7% of settled parent fields trace a connected route from goal to start: most failures are a valid route that differs from the labeled one, not a broken field. On Sudoku the gap is similar, with 96.9% of cells correct against 95.1% of grids. The residual also serves as a stopping rule: on S_5 at length 128, stopping once the largest per-site total-variation step falls below $5 \cdot 10^{-3}$ takes 81 steps on average against a cap of 256, and every test sequence is still correct.

4.4 Language-model results

Table 2 compares the FineWeb-Edu model with GPT-2. One Picard step is a weak language model on every task. Twenty-four steps, the depth used in training, raise ARC-Easy from 32.0% to 51.9%, SciQ from 37.4% to 74.4%, PIQA from 55.6% to 63.1%, and HellaSwag from 27.8% to 33.2%. The gain is not gradual: up to eight steps the scores stay at or below the one-step level, and most of the improvement arrives between K=8 and K=16. Training settles for 16 steps before its differentiated tail, and the beliefs become useful at about that depth. At $K = 24$ GFPR-LM is above GPT-2 small on all four tasks and above GPT-2 medium on ARC-Easy; it remains below GPT-2 medium on SciQ, PIQA, and HellaSwag. The comparison is not compute-matched: GPT-2 pays one trunk pass per token, while each Picard step is a full pass over the prompt and continuation. The scores are likelihoods of answer choices, not samples from the wavefront decoder of section 3.3. Held-out FineWeb-Edu validation perplexity is 18.9. The experiment is small by current standards: one 201M model, 6B tokens, and four likelihood-based benchmarks. We read it as a proof of concept rather than a competitive language model: the same belief-state update, with the output simplex enlarged to the vocabulary, trains into a working language model whose quality comes from iteration (protocol in section F).

5 Related Work

Recurrence in neural nets is older than the current puzzle models. Hopfield networks already iterate a state that is also the output: a binary pattern is updated until it matches a stored memory (Hopfield, 1982). Almeida–Pineda nets inject a constant input at every step and run until a fixed point (Almeida, 1987; Pineda, 1987); written as $h^{l+1} = f(h^l, x; w)$ this is depth recurrence with x clamped. Classic graph neural nets do the same on a graph, but they constrain the transition to be a contraction so the fixed point is unique, and they still decode with a separate head (Scarselli et al., 2009).

Weight-tied depth then reappeared as a way to spend test-time compute without growing the parameter count. The Universal Transformer ties one block across depth and can stop with ACT (Dehghani et al., 2019; Graves, 2016). Looped Transformers put a tied core between an untied prelude and coda, re-inject the input each step, and raise the loop count at test time (Giannou et al., 2023; Geiping et al., 2025). HRM nests two timescales and TRM collapses them to one stream; both can stop with ACT or a step cap (Wang et al., 2025; Jolicoeur-Martineau, 2025). The iterate in these models is a free latent, and a head turns the last state into symbols.

Deep equilibrium models treat an infinitely deep weight-tied stack as the root of $z = f_\theta(z; x)$ and differentiate at that root (Bai et al., 2019). FPRM and EqR use the same idea as the reasoning procedure itself (Movahedi et al., 2026; Huang et al., 2026). FPRM damps the Picard step and stops when the latent residual is small. EqR treats the map as an attractor landscape and can spend extra compute on restarts. In both, the state remains a vector in $\mathbb{R}^d$; a fixed point is not guaranteed by the architecture.

GFPR is an Almeida–Pineda-style depth recurrence: the input is injected every step, and the state is a field of categorical beliefs. As in a Hopfield network, the state is itself the output, but it is a learned distribution per site rather than a binary pattern matched to stored memories. The same state can carry structural constraints on a task (section 3.4), trained with a Fenchel–Young loss (Blondel et al., 2020).

On language, a standard Transformer has no depth recurrence: each block has its own weights and is visited once (Radford et al., 2019). Chain-of-thought spends compute by writing tokens (Wei et al., 2022; Kojima et al., 2022). Latent looped language models iterate a hidden state (Geiping et al., 2025). GFPR iterates a belief over the vocabulary at each site (section 3.3, section 4.4).

6 Conclusion

We argued that a looped reasoner should iterate its prediction rather than a latent code decoded at the end. In GFPR every intermediate state is a field of categorical beliefs that can be read as an answer, its residual measures how much that answer moves, and task structure can be imposed on it as a convex set. A fixed point then exists for any parameters, as a consequence of the design rather than its purpose; which one is reached is decided by training.

On Sudoku-Extreme, Maze-Hard, and S_5, this state is enough to exceed the published single-trajectory numbers of latent looped reasoners at similar scale, and on the puzzle of Figure 2 GFPR settles to a fixed point where FPRM's halt is not one. Adding task-specific convex structure helps where the relaxation is informative: the unit-flow readout raises Maze-Hard from 92.0% to 96.0%, while the Sudoku consistency readout so far only matches the product-of-simplices baseline. The language-model experiment is small, but it shows that the same update works when each site is a distribution over a vocabulary: after 24 Picard steps a 201M model trained on 6B tokens is competitive with GPT-2 small and medium on zero-shot multiple choice, and that quality appears only after iteration.

The evidence has clear limits. The baselines are quoted rather than re-run under a matched training recipe and inference budget, so the share of the gain due to the state space is not isolated. The language-model comparison is not compute-matched and scores likelihoods rather than generated text. A few Sudoku puzzles settle to a stable but wrong grid; because the state can be checked against the constraints without labels, restarts selected by constraint violation are a natural next step.

REFERENCES

Luís B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings of the IEEE First International Conference on Neural Networks*, volume 2, pp. 609–618, 1987.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922. doi: 10.4064/fm-3-1-133-181.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Mathieu Blondel, André F. T. Martins, and Vlad Niculae. Learning with Fenchel-Young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020.

L. E. J. Brouwer. Über abbildung von mannigfaltigkeiten. *Mathematische Annalen*, 71:97–115, 1911.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, pp. 41340–41391, 2025.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442, 2023.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. doi: 10.1073/pnas.79.8.2554.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning, 2026.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025.

C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in Applied Mathematics*. SIAM, 1995. doi: 10.1137/1.9781611970944.

Giancarlo Kerg, Kyle Goyette, Maximilian Puelma Touzel, Gauthier Gidel, Eugene Vorontsov, Yoshua Bengio, and Guillaume Lajoie. Non-normal recurrent neural network (nnRNN): learning long time dependencies while improving expressivity with transient dynamics. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35492–35506. PMLR, 2024.

Sajad Movahedi, Vera Milovanoviç, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers, 2026.

James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*, volume 30 of *Classics in Applied Mathematics*. SIAM, 2000. doi: 10.1137/1.9780898719468. Originally published 1970.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849, 2024.

Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1146. URL https://arxiv.org/abs/1905.05702.

Fernando J. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229–2232, 1987. doi: 10.1103/PhysRevLett.59.2229.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

Charlie Snell, Jaehun Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters, 2024.

Lloyd N. Trefethen and Mark Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, 2005.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

A FPRM AND GFPR SIDE BY SIDE

Figure 3 is one step of each method. The first three rows are the loop. The last row is not a further step. It is the reason a fixed point does or does not come for free.

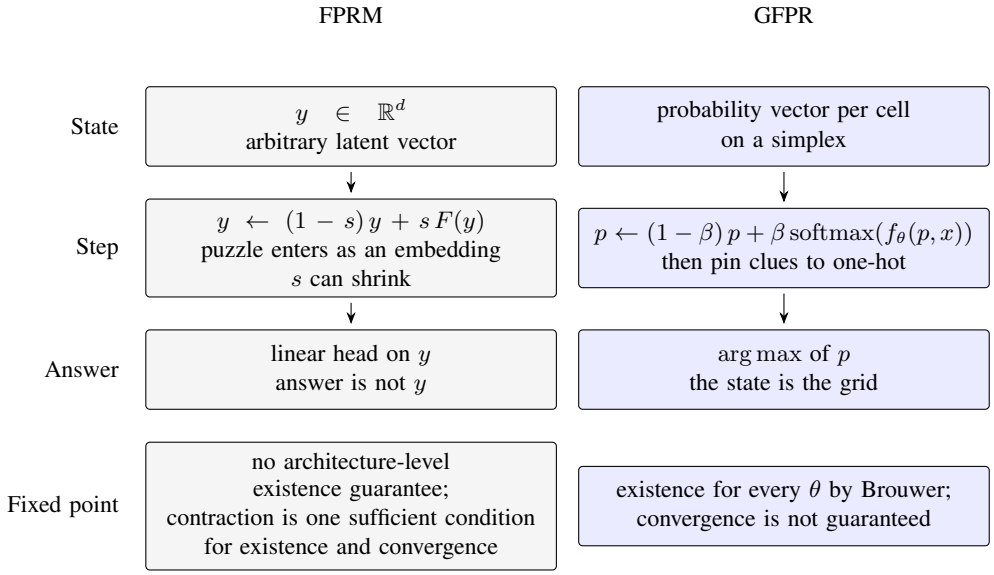

Figure 3: One test-time step. Arrows are the loop. The bottom row is a property of that loop, not another update. FPRM reads the grid from a head on an arbitrary latent. GFPR reads the grid from the belief it iterates, after pinning the given clues.

FPRM can stop when the step is small; GFPR runs for a fixed step budget at test time. They do not certify the same thing. In FPRM the step is motion of y. A head still turns y into digits; contraction is sufficient for a fixed point of F and for local convergence, but neither is guaranteed by the architecture. In GFPR the step is motion of p, and p is already the prediction. Clues stay pinned, the state stays in a compact product of simplices, so Brouwer gives a fixed point for every θ; which equilibrium the iteration reaches is still a training question.

B MODEL AND TRAINING SETTINGS

Table 3 lists the settings of the two released GFPR runs. Both use the same Transformer trunk over all sites. Inside one application of F_θ the trunk is run for a few self-conditioning passes: each pass takes the previous pass's softmax output as extra input, and the last one is the output of F_θ. Before each softmax the logits are soft-capped as $15\tanh(\ell/15)$.

The training rollout (Section 3.2 of the main text) has two phases. The first runs without gradients for at most D steps (random on Sudoku, fixed on Maze) and stops early once every puzzle in the batch has kept its maximum per-site TV step below a tolerance for two consecutive steps. On Sudoku, D is drawn per batch as $D = 1 + \mathrm{Poisson}(e^\tau)$ with $\tau \sim \mathcal{N}(\log \bar{D} - \sigma^2/2,\ \sigma^2)$, which has mean close to $\bar{D}$ and a heavy right tail; on Maze-Hard, D is fixed. The second phase differentiates a tail of m steps, and the loss is applied to its final state. With probability 0.25 a rollout starts from a random Dirichlet state instead of the uniform one. Evaluation and checkpoint selection use an exponential moving average of the weights.

Table 3: GFPR settings for the released runs. K is the number of output symbols and A the number of auxiliary coordinates; for S5, $K = 121$ counts the 120 permutations and the pad token. Rollout depth is the no-gradient prefix: "$\bar{D}$=32" is the log-normal Poisson draw in the main text, and the TV tolerance is its early-stop threshold. "cosine@150k" starts the learning-rate decay at step 150,000. S5 applies a causal depthwise convolution of kernel size 4 to the state before adding operation embeddings; the other runs do not.

	Sudoku-Extreme	Maze-Hard	S5
Parameters	7.16M	6.81M	5.60M
Width / layers / heads	256 / 9 / 8	512 / 2 / 8	224 / 9 / 8
K / A	9 / 16	6 / 8	121 / 16
Self-conditioning passes	4	2	4
Damping β	0.7	0.3	0.7
Rollout depth	$\bar{D}$=32	40	$\bar{D}$=32
Early-stop tolerance (TV)	$5 \cdot 10^{-3}$	10^{-5}	$5 \cdot 10^{-3}$
Differentiated tail	8	10	8
λ_{aux} / λ_{res}	0.02 / 0.02	0.02 / 0	0.02 / 0.02
Batch size	128	8	128
Learning rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
LR schedule	cosine@150k	cosine	cosine@150k
Warmup / steps	2k / 200k	1k / 60k	2k / 200k
Selected checkpoint	54k	42k	12k
Dropout / EMA decay	0.1 / 0.9999	0.1 / 0.9999	0.1 / 0.9999
Augmentation	Sudoku symmetries	dihedral	none

C Evaluation protocol

Sudoku-Extreme. Checkpoints are selected on a stratified 1,000-puzzle validation sample of the `test_hard` split of `sapientinc/sudoku-extreme-1k`; the remaining `test_hard` puzzles form a disjoint local test split for development. The selected checkpoint reaches 94.0% exact match on a 500-puzzle subsample of the validation slice with a 20,000-step budget. Headline evaluation is computed on all 422,786 puzzles of `sapientinc/sudoku-extreme` (`test.csv`)—a separate Hugging Face release from `sudoku-extreme-1k`—with the same cap as the released FPRM Sudoku-Extreme evaluation: a single trajectory from the uniform state, $\beta = 0.7$, and at most 35,000 damped Picard steps (early stop when every site's total-variation step falls below $5 \cdot 10^{-3}$ for two consecutive steps); that file is never used for checkpoint selection. Exact match is 95.11% and cell accuracy 96.93%.

Maze-Hard. Evaluation uses all 1,000 test mazes of `sapientinc/maze-30x30-hard-1k`, exactly 1,024 damped Picard steps, and $\beta = 0.3$. Exact match is 92.0%, cell accuracy 99.67%, and 99.7% of settled parent fields trace a connected route from goal to start.

S5. Training uses 2,000,000 sequences of 32 group updates, and evaluation uses all 1,000 test sequences of length 128. Inference runs at most 256 damped Picard steps at $\beta = 0.7$ from the uniform state and stops early when the maximum per-site total variation falls below $5 \cdot 10^{-3}$. On the selected checkpoint every test sequence is correct at every prefix length, so both sequence accuracy and final-state accuracy are 100%. The mean number of steps to that tolerance is 81.

Parent-field target. A breadth-first tree defines a unique parent only once the order in which neighbors are expanded is fixed, and most Maze-Hard instances have several shortest routes. We recover the expansion order of the dataset generator from the data and check that it reproduces every labeled route, so the traced route is compared with the same route that the published numbers score.

D Task-specific convex sets

This section gives the convex sets used by the constrained models in section 3.4.

Sudoku. For the Sudoku-Extreme dataset, let q_{rcd} denote the coordinate associated with digit $d \in \{1, \dots, 9\}$ at cell (r, c), and let $\mathcal{B}$ denote the collection of nine 3×3 boxes. For a puzzle x, let d_{rc} denote the given digit at each clue cell (r, c). We define

$$\mathcal{C}_{\mathrm{Sud}}(x) = \left\{ q \in [0,1]^{9\times 9\times 9} : \begin{array}{ll} \sum_{d=1}^{9} q_{rcd} = 1 & \forall r, c, \\ \sum_{c=1}^{9} q_{rcd} = 1 & \forall r, d, \\ \sum_{r=1}^{9} q_{rcd} = 1 & \forall c, d, \\ \sum_{(r,c)\in B} q_{rcd} = 1 & \forall B \in \mathcal{B}, d, \\ q_{rc,d_{rc}} = 1 & \text{for every clue cell } (r, c). \end{array} \right\}. \tag{9}$$

The first family of equalities makes $(q_{rc1}, \dots, q_{rc9})$ a probability vector for every cell. The remaining equalities require each digit to have total coordinate sum one in every row, column, and box, while the final constraints fix the clue coordinates. Every valid completed Sudoku grid corresponds to an integral point of $\mathcal{C}_{\mathrm{Sud}}(x)$. The converse does not hold for arbitrary feasible points because the polytope can contain fractional solutions; consequently, feasibility in $\mathcal{C}_{\mathrm{Sud}}(x)$ does not by itself imply a valid discrete Sudoku completion.

We use the power regularizer Ω_α defined in section 3.4 and define

$$q^*(z) = \operatorname*{arg\,max}_{q \in \mathcal{C}_{\mathrm{Sud}}(x)} \left\{ z^\top q - \Omega_\alpha(q) \right\}. \tag{10}$$

We evaluate Eq. equation 10 through its convex dual. After fixing the clue coordinates, let F denote the remaining coordinates and write the equality constraints as $A_F q_F = \widetilde{b}_x$. For a dual variable λ, define $s = z_F - A_F^\top \lambda$. Maximization of the Lagrangian with respect to the primal coordinates yields

$$q_i(\lambda) = \min \left\{ 1, \left[(\alpha - 1)[s_i]_+ \right]^{1/(\alpha-1)} \right\}. \tag{11}$$

The dual problem is solved numerically, and the resulting primal point is recovered from Eq. equation 11. Numerical feasibility is measured by the equality residual $\|A_F q_F(\lambda) - \widetilde{b}_x\|$.

The choice of α has only a modest effect on the number of recurrent iterations needed to reach the final accuracy: it shifts the transition point slightly, while the curves nearly coincide after sufficient iteration. The S5 and Sudoku ablations are reported in figure 4 and Table 4.

Maze. For Maze-Hard dataset, let $G_x = (V_x, E_x)$ be the directed graph whose vertices are the open cells and whose edge set contains both orientations of every adjacent pair of open cells. Let $B_x \in \mathbb{R}^{|V_x| \times |E_x|}$ be its node–edge incidence matrix, with coefficient $+1$ for an outgoing edge and -1 for an incoming edge. To match the parent-pointer decoding direction, we orient the flow from the goal g to the start s and define $b_x \in \mathbb{R}^{|V_x|}$ by

$$(b_x)_v = \begin{cases} 1, & v = g, \\ -1, & v = s, \\ 0, & \text{otherwise.} \end{cases}$$

The flow polytope $\mathcal{C}_{\mathrm{flow}}(x)$ is defined in equation 7. The equality $B_x f = b_x$ imposes flow conservation at every vertex other than g and s, together with net outflow one at g and net inflow one at s. The continuous relaxation may contain fractional flows and nonzero circulations. Therefore, membership in $\mathcal{C}_{\mathrm{flow}}(x)$ does not imply that f is the incidence vector of a simple path, nor does it imply shortestness or uniqueness. These properties are evaluated on the decoded discrete path separately.

Fenchel–Young training. Training uses the Fenchel–Young loss equation 8 and its gradient $\nabla_z L_{\Omega,\mathcal{C}}(z, y) = \Psi_{\mathcal{C}(x)}(z) - y$ when the argmax in equation 5 is unique (Danskin's theorem). The structured map can be applied either at every recurrent step or only at the terminal readout (section 3.4 lists which construction each benchmark uses). In the recurrent construction, the structured

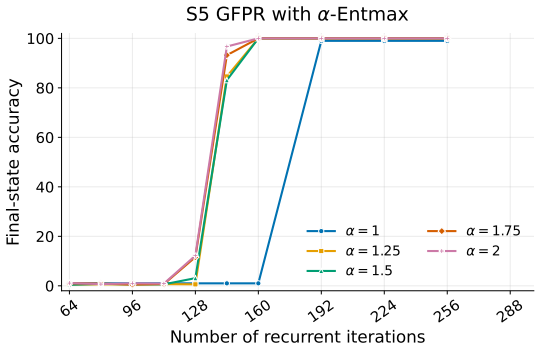

Figure 4: S5 final-state accuracy versus the number of recurrent iterations for different Entmax orders α.

α	Number of recurrent iterations					
	8	16	32	64	128	160
1.00	49.0	62.1	71.9	77.4	82.1	83.2
1.25	49.8	62.5	72.0	77.4	82.1	83.2
1.50	50.5	62.7	72.2	77.4	82.2	83.2
1.75	51.2	62.7	72.2	77.5	82.2	83.2
2.00	51.5	63.2	72.2	77.6	82.2	83.2

Table 4: Sudoku-Extreme exact-grid accuracy (%) for different Entmax orders α and recurrent iteration budgets.

component of every iterate belongs to $\mathcal{C}(x)$, and differentiation through an unrolled recurrence includes derivatives of the recurrent applications of $\Psi_{\mathcal{C}(x)}$. In the terminal-readout construction, the recurrence remains in the product-simplex state space of section 3.1, and $\Psi_{\mathcal{C}(x)}$ is applied only to the final score vector, so the gradient with respect to the final scores does not differentiate through the numerical iterations used to compute the structured prediction.

E SIMPLEX LANGUAGE-MODEL DECODE

Section 3.3 is implemented for a byte-pair TinyStories model in `talgat/lm/`. The trunk is a causal Transformer with weights tied across Picard steps. Training is the relax-plus-tail unroll of section 3.2: a non-differentiated settle, then a differentiated tail, with next-token cross-entropy on the output block of p_t against x_{t+1}. Open-ended generation uses wavefront decode (`generate_progressive` in `talgat/lm/code/decode_lm.py`). The defaults are soft-ahead width $W = 8$, damping $\beta = 0.5$, sharpening temperature $\tau = 1/16$, and a commit when either the top-two margin or a total-variation stability test fires, after a minimum number of Picard steps since the previous commit. There is no key–value cache: every Picard step is a full forward over the active length, so iteration counts describe this decoder rather than a cached autoregressive Transformer. The FineWeb-Edu multiple-choice scores in section 4.4 use fixed-K teacher-forced unrolling, not wavefront sampling.

F FINEWEB-EDU LANGUAGE-MODEL EVALUATION

The 201M run in section 4.4 is trained on 6,000,148,480 FineWeb-Edu tokens for 36,622 steps. The trunk has width 2048, 16 heads, and two weight-tied blocks; the vocabulary has 16,384 byte-pair tokens and 16 auxiliary coordinates; the context is 512. Training uses $\beta = 0.5$, 16 no-gradient steps and an 8-step differentiated tail, and Muon with AdamW. Reported scores use EMA weights.

Zero-shot scoring is next-token log-likelihood of each answer choice, with a BOS token, full continuation, and left truncation only on overflow. The state starts uniform; the iteration count K is fixed (no residual halt). Headline metrics are accuracy on ARC-Easy (test, N=2376), SciQ (test, N=1000), and PIQA (validation, N=1838), and length-normalized accuracy on HellaSwag (validation, N=10042). Final arithmetic is FP32 states and activations with TF32 matmul.

Table 5 is the official depth sweep for this checkpoint. Most of the gain appears between K=8 and K=24. The step K=24 $\rightarrow$ 32 does not improve all four tasks, which is why the main text uses the training depth K=24. These scores do not show that the simplex state has reached a fixed point.

Table 5: GFPR-LM 201M zero-shot accuracy (%) versus Picard depth K. HellaSwag is length-normalized accuracy.

Task	K=1	2	4	8	16	24	32
ARC-Easy	32.0	32.1	29.4	28.4	49.5	51.9	51.9
SciQ	37.4	34.6	29.9	32.9	76.1	74.4	74.3
PIQA	55.6	55.6	54.5	53.8	62.8	63.1	63.2
HellaSwag	27.8	27.6	26.8	26.0	32.2	33.2	33.1

GPT-2 small and medium in Table 2 of the main text use one forward pass, batch 16, bfloat16, in the same task configurations. GPT-2 large (774M) on the same four tasks is 53.0 / 79.7 / 70.1 / 45.3; we omit it from the main table because it is several times larger than the 201M GFPR-LM run.

G FPRM DIAGNOSTIC PROTOCOL

The traces in the residual-comparison figure of Section 2 of the main text use FPRM's public Sudoku checkpoint and its evaluation loop, run in float64 for 80 steps on one test puzzle, with step-size decay 0.997 and patience 10 as in the released configuration. GFPR is run on the same puzzle with $\beta = 0.7$ from the uniform state. The residual is the RMS ratio $\|F(u) - u\|/\|u\|$ at every step.

Local amplification probe (power iteration). At state u we set $J_F = \partial F/\partial u$ and access it only through Jacobian–vector products from reverse-mode autodiff on F. With a unit probe v_0, for $t = 1, \ldots, K_{\text{pi}}$ we compute $w_t = J_F v_{t-1}$, normalize $v_t = w_t/\|w_t\|_2$, and report the probe value $\widehat{\alpha}(u) = \|w_{K_{\text{pi}}}\|_2$. For a diagonalizable J_F with a simple dominant eigenvalue, $\widehat{\alpha}(u)$ approaches $|\lambda_{\max}(J_F)|$ as $K_{\text{pi}} \to \infty$; for a non-normal J_F, finite K_{pi} can reflect transient amplification and need not certify $\rho(J_F)$ (Trefethen & Embree, 2005). The right-hand trace in the main-text figure plots $\widehat{\alpha}(u_k)$ at each iterate u_k with K_{pi}=20 (float32). At FPRM's halt (step 36) the residual is $3.2 \cdot 10^{-2}$ and $\widehat{\alpha} \approx 1.03$; after the halt the residual stays between $1.3 \cdot 10^{-2}$ and $3.1 \cdot 10^{-2}$. GFPR's residual falls below 10^{-12} by step 27, with $\widehat{\alpha} \approx 1.4 \times 10^{-3}$ at the final step of the trace. On the final states of this one puzzle we cross-check with restarted Arnoldi on the flattened Jacobian (Krylov dimension 35, 3 restarts; exact JVPs for FPRM, finite-difference products for GFPR). The Ritz values are not the full spectrum, but they approximate its outer part. For FPRM near its halt (step 33 in this run) the largest Ritz values are the pair $-0.126 \pm 1.031i$ ($|\lambda| \approx 1.039$), followed by real values near -0.98; no Ritz value has real part outside $(-1, 1)$. The pair becomes contracting for the damped map only when $s < 2(1 - \operatorname{Re}\lambda)/|1 - \lambda|^2 \approx 0.966$; the step size at halt is at least $0.997^3 \approx 0.991$, giving $|1 - s + s\lambda| \approx 1.03$. For GFPR the largest Ritz value is $1.34 \cdot 10^{-3}$.

The population audits use held-out validation puzzles. Halting statistics come from 1000 puzzles run with the authors' loop up to 35000 steps (median halt at step 126, median step size at halt 0.965). The population audit applies the same probe on 128 puzzles after 8000 iterations in float32 with halting disabled, with K_{pi}=10 at the final state; 75% have $\widehat{\alpha}(u) > 1$. We have not run Arnoldi on these states, so this fraction is not a certified statement that $\rho(J_F) > 1$ on 75% of puzzles.